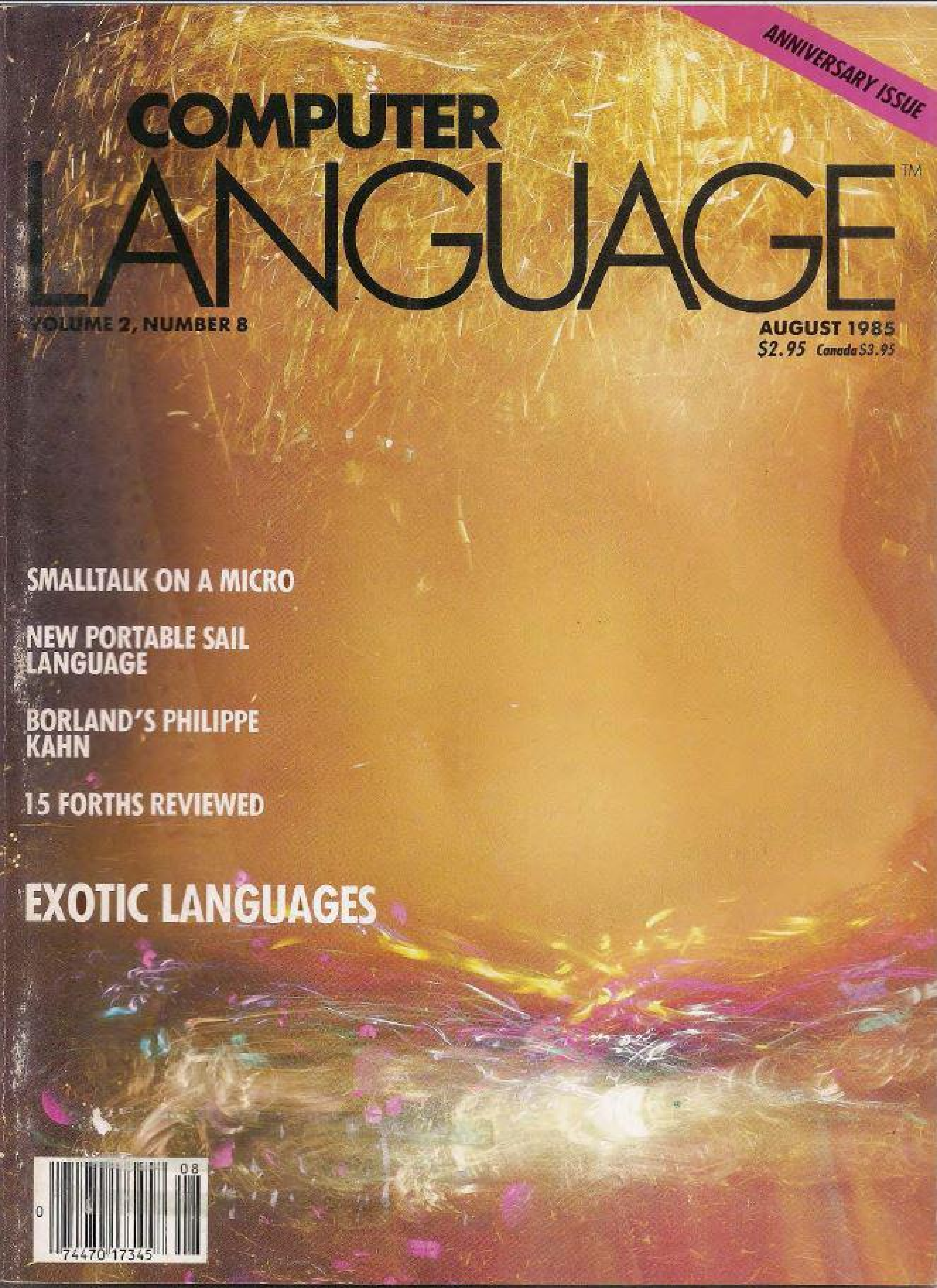


ANNIVERSARY ISSUE

COMPUTER LANGUAGE™

VOLUME 2, NUMBER 8

AUGUST 1985
$2.95 Canada $3.95

SMALLTALK ON A MICRO

NEW PORTABLE SAIL LANGUAGE

BORLAND'S PHILIPPE KAHN

15 FORTHS REVIEWED

EXOTIC LANGUAGES

08
0 74470 17345

YOU'VE GOT THE BEST PASCAL COMPILER! NOW — GET THE BEST UTILITY!

TURBO SCREEN™

$49.95

NEVER AGAIN WRITE SOURCE CODE FOR SCREEN DISPLAYS!

*If you **LIKE** Turbo Pascal*, you'll **LOVE** TURBO SCREEN™!*

Tired of writing line after line of source code just to create effective screen displays and error-proof data handling? **Then** use TURBO SCREEN's Editor to create the screens, the Collator to define a list of screens . . . and then **relax** for a few seconds **while the Generator writes the code!**

TURBO SCREEN™

- 100 Fields per screen, and up to 80 screens in your application.
- One screen or eighty, the size of your program doesn't change.
- I/O field types of:
  Real, Integer, String, Character, Boolean.
- "Bullet-proof" data entry.
- Create Window-Style overlays or Full-screen pictures in CP/M*, MS-DOS*, or PC-DOS.
- Supports video attributes for your terminal. And YES, if you have an RGB monitor, you can create screens in COLOR on your IBM PC or true compatible.
- A SINGLE LINE of source code invoking TURBO SCREEN'S "display" procedure controls:
  - —picture selection
  - —output to screen, printer, or disk
  - —I/O field update
- TURBO SCREEN™ is completely menu-driven and includes a built-in Screen Editor, Collator, and Generator, each called up with a single keystroke!
- ADVANCED EDITOR:
  - —Turbo Pascal*—like commands include:
    - —Block commands for copy, fill, exchange, erase.
    - —Draw lines in any direction with any character.
    - —Supports IBM color monitor and graphics characters.
- FAST—Generates code for 20 screens in about 60 seconds!
- DISK UTILITIES built-in:
  - —directories
  - —erase files
- REQUIRES:
  - —Turbo Pascal any version
  - —80x24 or larger video screen
- AVOID software "bottlenecks!"

PASCOM COMPUTING

23611 Chagrin Blvd., Suite 101
Cleveland, Ohio 44122

Check ______________

Money Order ______________

Visa ______________

Master Card ______________

Card # ______________

Exp. Date: ______________

TURBO SCREEN™	
package	$49.95
Plus Ship. (UPS)	5.00
Total	$54.95

Start letting TURBO SCREEN™ write your I/O source code today!

VISA MasterCard ONLY — Call TOLL-FREE: 1-800-243-1849

Inside Ohio call 1-216-292-8745 (Lines Open 24 hours, 7 days)

Computer System: ______ 8-bit ______ 16-bit

Operating System: ______ CP/M80 ______ PC-DOS

______ CP/M86 ______ MS-DOS

Computer Model: ______________ Disk Format: ______________

Name: ______________

Address: ______________

City: ______________ State ______ Zip ______

Telephone: ______________

Ohio residents add 6½% sales tax. Outside U.S.A. add $20.00

U.S. Dealer Inquiries Welcome.

*Turbo Pascal is a trademark of Borland International. IBM is a trademark of International Business Machines. MS-DOS is a trademark of Microsoft. CP/M is a trademark of Digital Research.

Software development isn't a mountainous task once you eliminate the high C errors.

When you can find and fix bugs at the earliest possible moment, creating software stops being such an uphill grind.

And the Smart/C Environment makes it possible. It's a complete, fully-integrated development environment for C that saves you from the creativity-inhibiting cycle of edit, compile, re-edit, re-compile, link, load, test, re-edit, re-compile, etc., ad infinitum. Smart/C puts the fun back in programming, because you spend your time creating... not waiting.

Here's why. Syntax errors are eliminated automatically as code is entered. Smart/C's highly integrated editor and interpreter allow you to interpret your program at any time in the creation process, so logic errors can be ferreted out as soon as the algorithm exists—long before any compile, link, or load.

The complete integration of the editor and interpreter means you can stop anywhere in the interpret cycle, edit, and then go right back into the interpreter exactly where you left off. Not only that, the screen-oriented user interface lets you see all operations, even interpretation, right on the listing of the code.

And to make maintenance programming easier, Smart/C's Migrator allows existing C code produced with any editor to be modified and run within the Smart/C Environment.

All of which makes Smart/C an excellent tool. It's flexible, non-restrictive, and lets you create elegant, readable, error-free programs that you can watch run with a great feeling of satisfaction.

Smart/C™

Free Demo Disk!

To fully appreciate Smart/C, you have to see it in action. For your free IBM PC MS-DOS demo disk, call us. Or write us on your company letterhead.

AGS Computers, Inc., Advanced Products Division,
1139 Spruce Drive, Mountainside, NJ 07092.
800-AGS-1313. In NJ, 201-654-4321.

Smart/C Features

The Smart/C Environment
- ☐ Fully integrated editor and interpreter
- ☐ Only one load brings them both in
- ☐ One command set
- ☐ Move between one another at will

Syntax Directed Editor
- ☐ vi-like command set
- ☐ Automatically provides formats for blocks, *for, case* and *if* statements

Interpreter
- ☐ Current module can call external modules during interpretation
- ☐ Has Include capability
- ☐ Totally precompilation—no incremental compile
- ☐ Can interpret partially defined files allowing for rapid prototyping
- ☐ Variable speed of interpretation
- ☐ Multiple windows with user-defined sizes

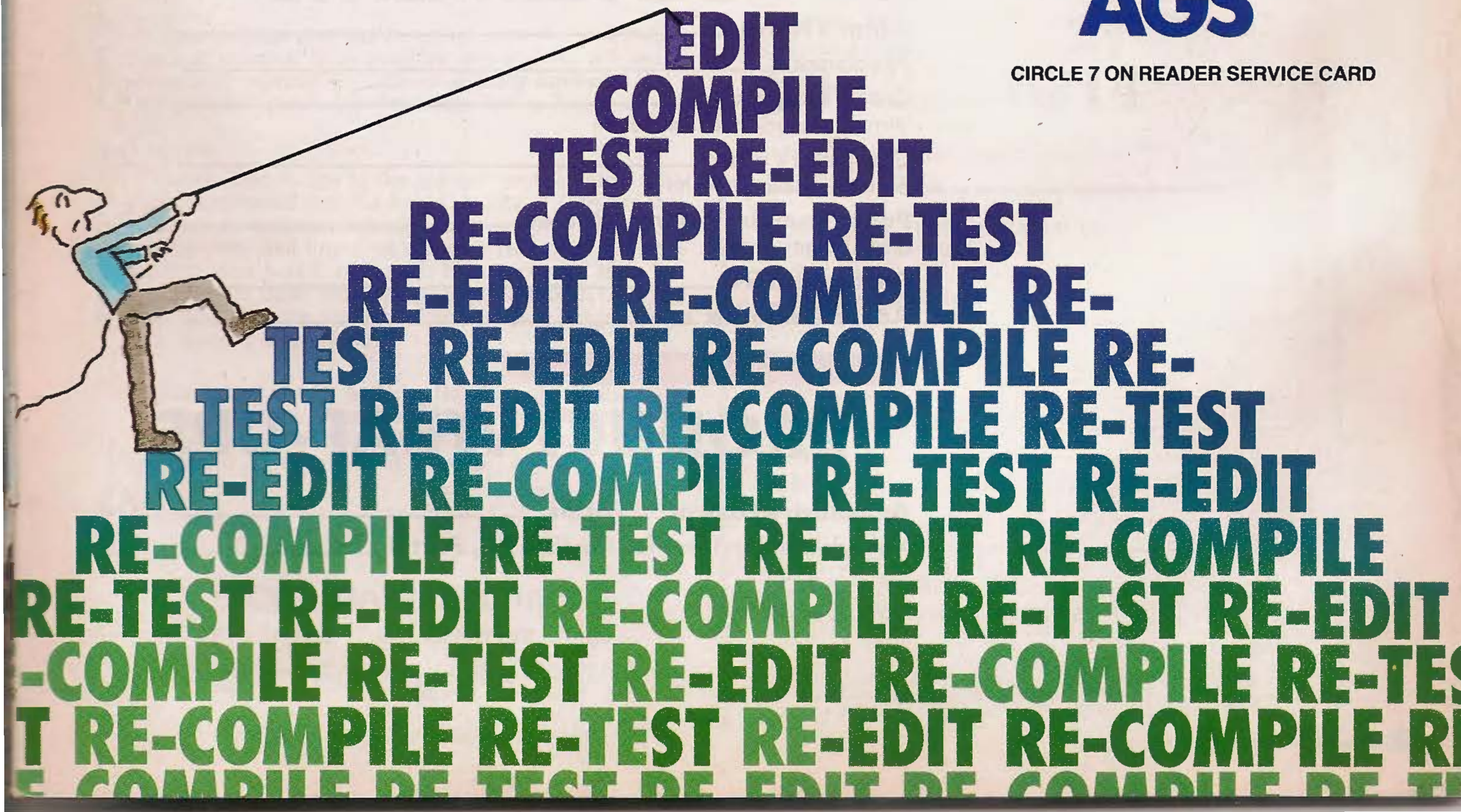
The Smart/C Migrator
- ☐ Allows C code produced with any editor to be interpreted by Smart/C
- ☐ Reformats for readability

Smart/C has been ported to UNIX™ System V Release 2, Berkeley 4.2, Xenix,™ and MS-DOS. Versions run on 8086- and 68000-based machines, as well as proprietary architectures. Smart/C runs on PCs, micros, supermicros, minis, and even mainframes.

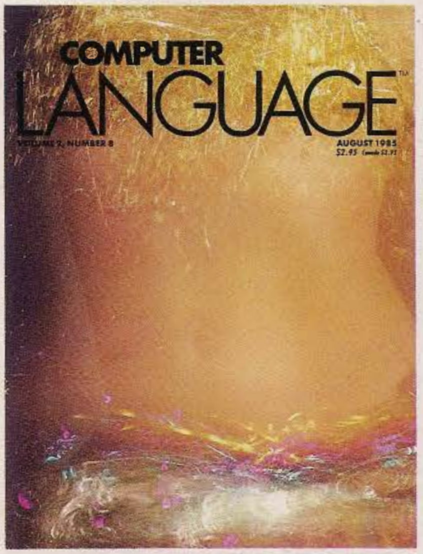
Trademarks—Smart/C: AGS Computers, Inc.; UNIX: AT&T Bell Labs; Xenix and MS-DOS: Microsoft Corp; IBM PC: IBM Corp.

AGS

CIRCLE 7 ON READER SERVICE CARD


EDIT
COMPILE
TEST RE-EDIT
RE-COMPILE RE-TEST
RE-EDIT RE-COMPILE RE-
TEST RE-EDIT RE-COMPILE RE-
TEST RE-EDIT RE-COMPILE RE-TEST
RE-EDIT RE-COMPILE RE-TEST RE-EDIT
RE-COMPILE RE-TEST RE-EDIT RE-COMPILE
RE-TEST RE-EDIT RE-COMPILE RE-TEST RE-EDIT
-COMPILE RE-TEST RE-EDIT RE-COMPILE RE-TE
T RE-COMPILE RE-TEST RE-EDIT RE-COMPILE R

COMPUTER LANGUAGE



ARTICLES

Smalltalk Comes to the Micro 27
by Darryl Rubin
From the research labs of Xerox Corp. to today's micro comes Smalltalk—the pioneering software development language that helped make object-oriented programming, message passing semantics, and modeless design a reality.

BLISS Lures Systems Programmers 35
by Bruce Leverett
Comparable with C and other systems-oriented languages, BLISS demonstrates how programmers easily can be lured away from using assembly language by means of a high-level language with equal capabilities.

PSAIL: SAIL to C 39
by Peter F. Lemkin
The SAIL language was created at Stanford University's AI Lab for writing large, complex application and system programs. Now you can have the portable public domain source code to a new language called PSAIL, which brings all the power of SAIL to most medium-size C environments.

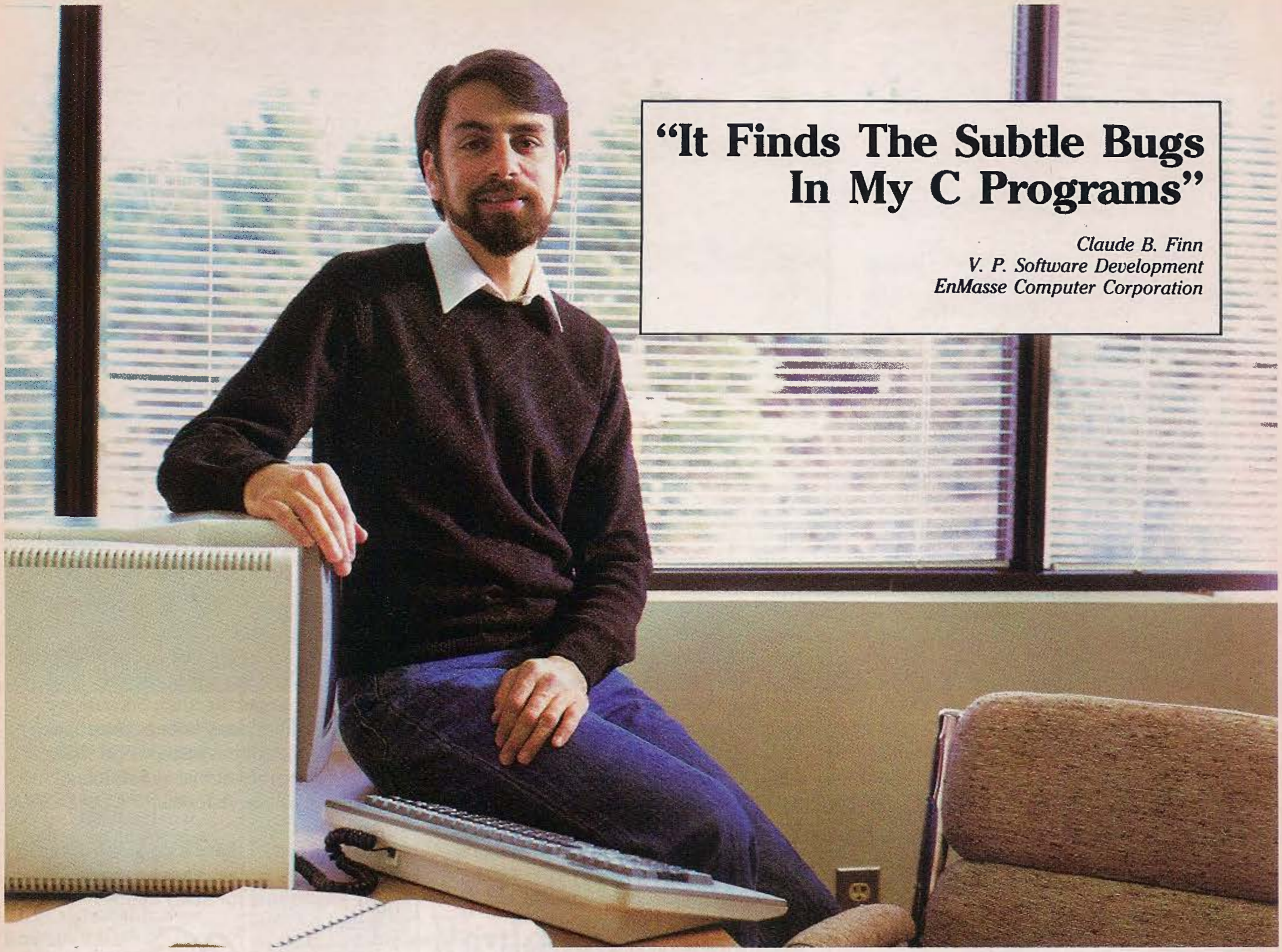
Logic at a Glance, Part II 49
by Jim McCarthy
Part II of this series on the ERGO Logic Kit focuses on those parts of the kit that provide facilities for the creation, editing, and compilation of decision tables. The kit's rationale and philosophy is described in some detail.

DEPARTMENTS

Editor's Notes 4
Feedback 7
CrossThoughts 11
Virtual memory data simulation
ComputerVisions 21
Borland's Philippe Kahn
Public Domain Software Review 57
dBASE II programs
Product BINGO 61
Advertiser Index 104

PRODUCT WRAP-UPS

An Industry Look at 15 Forths 63
Knowledge Systems for the IBM PC, Part II 87



"It Finds The Subtle Bugs In My C Programs"

Claude B. Finn
V. P. Software Development
EnMasse Computer Corporation

The SAFE C™ Family Can Literally Cut Software Development Time In Half. For UNIX™ and VAX/VMS.™

"Evasive bugs that use to eat up days — I'm finding them in minutes. Stray pointers, errant array indexes, parameter mismatches, misuse of string functions...I'm using Safe C automatic error detection to find them all."

Claude Finn is one of the many C programmers who have discovered that the Safe C family of software development tools dramatically enhances programmer productivity and improves software reliability and portability. Most Safe C customers have recouped their investment in these tools within the first month of active use. And with the security of Safe C their programmers are sleeping a lot easier!

The Safe C Family includes the Runtime Analyzer, Dynamic Profiler, Standalone Interpreter, English to C Translator and C to English Translator. They may be purchased separately or as a group.

CCA Uniworks, Inc.
Productivity Tools for Programmers
20 William Street • Wellesley, MA 02181

CIRCLE 60 ON READER SERVICE CARD

For more information or to place an order call our customer representatives at

800-222-0214

In MA (617) 235-2600
or mail this request form today.

Please send me information on:

☐ The Safe C Development Tool Family
☐ AI Development Tools
☐ CCA EMACS Unix/VMS Editor Environment
☐ Your complete line of state-of-the-art programming tools
☐ Please send license forms

Name ______________________
Title ______________________
Company ______________________
Address ______________________
City, State, Zip ______________________
Phone (   ) ______________________

CCA UNIWORKS, INC.
20 William Street • Wellesley, MA 02181

A Crowntek Company

Unix is a trademark of Bell Laboratories, VAX and VMS is a trademark of Digital Equipment Corporation, MS/PC-DOS is a trademark of Microsoft, Safe C is a trademark of Catalytix Corporation, CCA EMACS is a trademark of CCA Uniworks, Inc.

CL885

COMPUTER LANGUAGE

EDITOR
Craig LaGrow

MANAGING EDITOR
Regina Starr Ridley

TECHNICAL EDITOR
John Halamka

PRODUCT REVIEW EDITOR
Hugh Byrne

EDITORIAL ASSISTANT
Kathy Kincade

CONTRIBUTING EDITORS
Doug Millison, Tim Parker,
Namir Clement Shammas

SPECIAL PROJECTS MANAGER
Jan Dente

ADVERTISING COORDINATOR
Mary Gordon

OPERATIONS CONSULTANT
Beatrice C. Blatteis

CIRCULATION COORDINATOR
Renato Sunico

ART DIRECTOR
Jeanne Schacht

COVER PHOTO
Dow/Clement Photography

PRODUCTION ARTISTS
Anne Doering, Barbara Luck

PRODUCTION
Steve Campbell, Kyle Houbolt

TECHNICAL CONSULTANT
Addison Sims

ACCOUNTING MANAGER
Lauren Kalkstein

WHOLESALE COORDINATOR
Nicola Sullivan

PUBLISHER
Carl Landau

COMPUTER LANGUAGE *(ISSN 0749-2839) is published monthly by* COMPUTER LANGUAGE *Publishing Ltd., 131 Townsend St., San Francisco, CA 94107. (415) 957-9353.*

Advertising: For information on ad rates, deadlines, and placement, contact Carl Landau at (415) 957-9353, or write to: COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

Editorial: Please address all letters and inquiries to: Craig LaGrow, Editor, COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

Subscriptions: Contact COMPUTER LANGUAGE*, Subscriptions Dept., 2443 Fillmore St., Suite 346, San Francisco, CA 94115. Single copy price: $2.95. Subscription prices: $24.95 per year (U.S.); $30.95 per year (Canada and Mexico). Subscription prices for outside the U.S., Canada, and Mexico: $36.95 (surface mail), $54.95 (air mail)—U.S. currency only. Please allow six weeks for new subscription service to begin.*

Postal information: Second-class postage is paid at San Francisco, CA and at additional mailing offices.

Reprints: Copyright 1985 by COMPUTER LANGUAGE *Publishing Ltd. All rights reserved. Reproduction of material appearing in* COMPUTER LANGUAGE *is forbidden without written permission.*

Change of address: Please allow six weeks for change of address to take effect. POSTMASTER: Send change of address (Form 3579) to COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, CA 94107.*

COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, CL Publications. All material published in* COMPUTER LANGUAGE *is copyrighted © 1985 by CL Publications, Inc. All rights reserved.*

Editor's Notes

In the days when computers were first being designed, the challenge of programming was almost insurmountable because of one unbelievable truth: programming languages did not yet exist. Then only the computer's native code was available, and using it was a painstakingly detailed process.

But today—like the roots, limbs, and branches of a tree—we not only have languages, but hundreds of languages and variations upon variations of languages. The entire arena of programming languages is a marvelously rich and diverse field.

It is amazing how many of the older, less well-known languages in the world have survived (many without the support of any major computer manufacturer) and how they continue to please groups of people who make exceptionally strong demands on their computing languages. Many of these so-called exotic languages have sprung up between the cracks and have helped us discover the wealth of detail and complexity found in real programming languages.

This month's special issue takes a look at some of the relatively unknown programming languages out there which are used in small academic centers, as in-house systems languages, or as home-grown applications-oriented languages. People who read *COMPUTER LANGUAGE* apparently like to see material on this subject because our Exotic Language of the Month Club department has been our hottest mail generator.

While no anthology could possibly cover all of the existing small languages, *COMPUTER LANGUAGE* will attempt to compile a data base of information on this industry phenomenon. Over time, we plan to compile a comprehensive directory and try to convey historically how each language interrelates.

So if you're aware of a particular language that you think deserves coverage, please speak up!

This issue is also a special occasion for all of us here at *COMPUTER LANGUAGE*—it's our one-year anniversary! Let me extend a heartfelt thanks to all those who have supported us over the past year.

We've been getting the kind of feedback that suggests *COMPUTER LANGUAGE* is maturing more and more as each issue goes by. The issues are becoming easier and easier to put together, because we have attracted many talented free-lance writers who submit manuscripts that reflect the professional and technical tone we originally envisioned for *COMPUTER LANGUAGE*.

We, the staff, have a lot of fun doing what we do, and we look forward to having you around for many anniversaries to come!

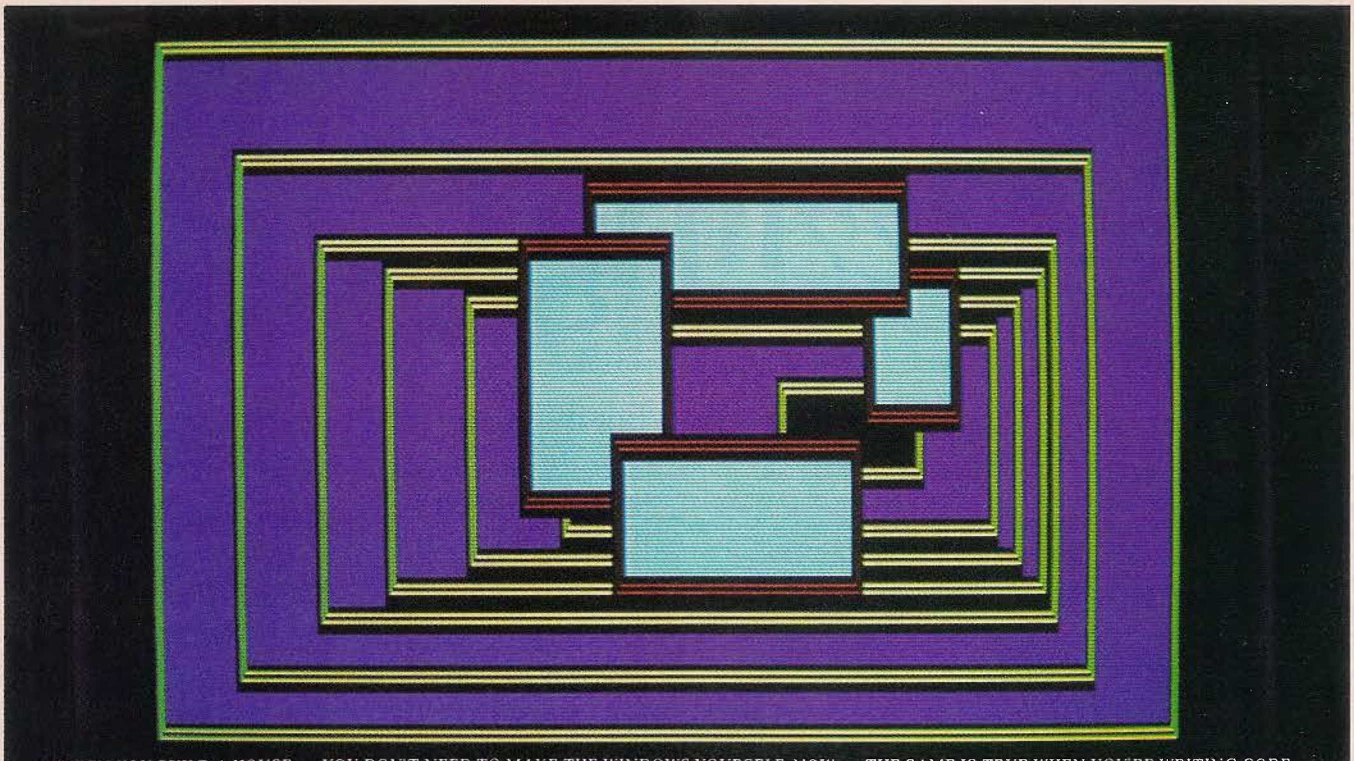
Craig LaGrow
Editor

How to get COMPUTER LANGUAGE source code listings

All the source code mentioned in the magazine can be obtained in one of three ways:

- Two bulletin board systems. To upload and download text and binary programs, call (415) 957-9370 at 300/1200 baud—24 hrs.
- CompuServe Special Interest Group. Type "GO CLM" at any prompt.
- *COMPUTER LANGUAGE* Users Group. $6.50 for every 5¼-in. disk. $8.50 for 8-in. SSSD. Determine, by the table below, how many disks you will need and specify your disk format. Mail your request and check to our office (address at left).

Vol.	Issue	Size
1	Premier '84	130K
2	Oct. '84	42K
3	Nov. '84	150K
4	Dec. '84	140K
5	Jan. '85	40K
6	Feb. '85	190K
7	Mar. '85	230K
8	Apr. '85	230K
9	May '85	150K
10	June '85	80K
11	July '85 (not incl. ERGO)	100K
12	July '85 (ERGO Logic Kit)	500K
13	All Code Swap Shops to date	350K
14	Aug. '85	150K



WHEN YOU BUILD A HOUSE . . . YOU DON'T NEED TO MAKE THE WINDOWS YOURSELF. NOW . . . THE SAME IS TRUE WHEN YOU'RE WRITING CODE.

Windows With A View Toward The Future

The Window Machine™ occupies only 12K! Written in tight, fast Assembler, it performs like a racing engine...with more power than you'll probably ever need. Yet, it's an engine designed to fit in the vehicle of your choice...from a "stripped-down" 128K IBM PC to a fully loaded AT. The programs you write today will run on the broadest range of machines possible... now, and in the future.

These are coders' windows... designed to be built into the programs you are writing. They can overlap, move anywhere on the screen, grow, shrink, vanish or blink. They can be bordered in anything from a simple line to flashing asterisks...or even no border at all. And you can have up to 255 of them at a time! Color or monochrome ...of course!

Windows Bigger Than Your Screen?

Here's where the VSI part of our name fits in. VSI means Virtual Screen Interface. Behind each window, there's a much bigger picture. VSI defines virtual screens rather than just windows. The window itself shows whatever portion of its virtual screen you wish to exhibit at any given point in your program. Each screen can be up to 128 x 255 (columns x rows, or rows x columns). And there are more than 100 screen primitives at your command.

Multilingual Windows

You can order The Window Machine with the language interface of your choice: C, Pascal, Compiled Basic, Fortran, Cobol, or PL1. We've even recently completed an interface for Turbo Pascal*, so that now true, full-featured windowing can be utilized with this fine compiler. (Turbo's own built-in "windowing" procedure is extremely limited).

Why did Simon & Schuster, 3Com, Tymshare, and Revlon choose VSI–The Window Machine?

(and how come you can buy it for such a low price?)

Windows That Won't Break You

We decided to save you a lot of money. So, we left behind fancy binders, monogrammed slip cases and plastic presentation boxes. Instead, you'll find an extremely powerful tool and a 200 page manual written with an eye toward simplicity, clarity and completeness. (We figured if you wanted ribbons and bows you could always add them yourself.)

And by offering you the product ourselves, we were able to cut out all the middlemen and save you a tremendous amount of money.

VSI *THE WINDOW MACHINE*

Available for the IBM PC, XT, AT, IBM Compatibles, Wang, T.I., and HP 150

The Window Machine Includes:

$59.95

- *Zoom Windows*
- *Multiple Virtual Screens (up to 255)*
- *Choice of Borders (including flashing borders)*
- *Support for all Color and Monochrome Video Attributes (no graphics card required)*
- *Built-in Diagnostics*
- *And much, much more*

ORDER YOUR COPY OF VSI—THE WINDOW MACHINE TODAY

For Visa & MasterCard orders call toll free:

800-538-8157 Ext. 824 In CA 800-672-3470 Ext. 824

Call Mon.-Fri. 6A.M. to 12P.M., Sat. & Sun. 6A.M. to 8P.M. (P.S.T.)

The Window Machine™ $59.95 + $5 Shipping and Handling

LANGUAGE INTERFACE:

☐ Lattice C ☐ Realia Cobol ☐ Microsoft Basic Compiler ☐ Microsoft Fortran ☐ PL1 ☐ Microsoft Pascal ☐ Turbo Pascal (full featured true windowing)

COMPUTER ____________________

Name ____________________

Address ____________________

City ____________________ State ____ Zip Code ________

☐ Check ☐ Money Order ☐ VISA ☐ MasterCard

Card # ____________________ Exp. Date ________

*California residents: tax included. Orders outside USA: Please add $10 for shipping and handling

30 day Money Back Guarantee

CL

AMBER
AMBER SYSTEMS
1171 S. Saratoga-Sunnyvale Road
San Jose, CA 95129

*Turbo Pascal is a Trademark of Borland International

AMBER SYSTEMS, INC. 1171 S. Saratoga-Sunnyvale Road, San Jose CA 95129

FOR DEALER INQUIRIES: CALL OUR 800 NUMBER

CIRCLE 2 ON READER SERVICE CARD

New Version

Sizzling C.

The fastest C. The C that Microsoft developed to write its own software programs. Hot.

So hot that we can make this claim: Virtually every program runs faster with Microsoft® C Compiler than with any other MS-DOS® C compiler.

Efficient C.

We give you everything you need to write code so tight your computer will scream.

"Preliminary testing on the Microsoft C Compiler produced code that was significantly smaller than that produced by other C compilers."
Paul Springer, Ashton-Tate.

"We found the FAR pointer very helpful for situations where a mix of memory models offers the greatest efficiency."
Robert Frankston, Software Arts.

"The portability of the code between MS-DOS and XENIX® is great."
Jim Bean, Peachtree Software.

But it's really no surprise that our C stretches your micro to its limits. We wrote both the MS-DOS and the XENIX operating systems.

For the name of your nearest Microsoft dealer, or to upgrade from Microsoft C Compiler or Lattice C, call (800) 426-9400. In Washington State, Alaska, Hawaii and Canada, call (206) 828-8088.

MICROSOFT.®
The High Performance Software™

And make your programs really cook.

Microsoft C Compiler Version 3.0

Microsoft C Compiler
- Produces compact code and fast executables.
- Implements register variables.
- Small, Medium and Large Memory model Libraries-Mix models with NEAR and FAR pointers.
- Transport source and object code between MS-DOS & XENIX operating systems.
- Library routines implement most of UNIX System V C library.
- Choose from three Math libraries and generate in-line 8087/287 instructions or floating point calls.
  - Floating point emulator (utilizes the 8087/287 if installed).
  - 8087/287 coprocessor support.
  - Alternate math package—extra speed on systems without an 8087/287.
- Link routines written in Microsoft FORTRAN (V 3.3 or higher), Microsoft Pascal (V 3.3 or higher) or Microsoft Macro Assembler.
- Supports MS-DOS pathnames and Input/Output redirection.
- File sharing and record and file locking is supported.
- Do source level debugging, with the Symbolic Debug Utility, available separately with the Microsoft Macro Assembler Package.*

Library Manager
Create, organize and maintain your object module libraries created with Microsoft languages.

Object Code Linker
- Simple overlay linker combines relocatable object modules created using Microsoft Languages into a single program.
- Link very large programs (over 1MB, using overlays).

EXEPACK Utility
A new utility to compress sequences of identical characters from an executable file and to optimize the relocation table.

EXEMOD Utility
A new utility used to modify the fields in the header according to the instructions given by the user in the command line.

'C' Benchmarks—done on a Compaq Plus with 512k memory with no 8087. Program "SIEVE," with register variables.

	Exec Time	Code Size	EXE Size
Microsoft C	:9.39	141	5,914
Lattice C	:12.24	164	20,072

***Purchase both Microsoft C Compiler and Microsoft Macro Assembler and get a $25 rebate direct from Microsoft. See package for details.**

Microsoft, MS-DOS and XENIX are registered trademarks and The High Performance Software is a trademark of Microsoft Corporation.

FEEDBACK

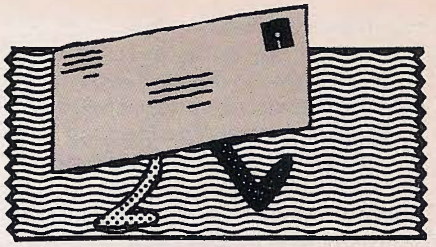


Illustration: Anne Doering

Recursing Forth

Dear Editor:

Jean-Pierre Schachter's article on recursion in Forth (*COMPUTER LANGUAGE*, June 1985, pp. 27-30) was very interesting. Recursion is a useful technique, and as the author states, is easy to do in Forth.

A word of caution, however. Forth has two stacks, the parameter stack (used for calculations and the passing of data) and the return stack. When a Forth verb calls another verb, an address is pushed on the return stack. The maximum size of the return stack is not unlimited, of course, so if a recursive procedure nests sufficiently deeply, the return stack will fill. This error is not detected in most Forth implementations since the speed penalty would be too great. So most Forths will eventually crash if the stack grows too large.

To check to see how many levels your return stack can handle, try the (appropriately named) verb *CRASH* (Listing 1). This verb is an endless recursion with a nest count. It will print 2, 3, 4, . . . until Forth crashes. (It begins with 2 because the *RECURSE* verb is nested once.) The last number it prints is the number of levels that your return stack can handle.

This will only give you a rough idea, of course, since different Forths have different things below the stack. In the standard fig-Forth memory map, the terminal input buffer (TIB) is below the stack. In that case, the return stack might be able to borrow the TIB without crashing.

The | (*BAR*) verb listed by Dr. Schachter works very well but has one potential tial danger (though not a serious one). If it is used only once (or an odd number of times) in a definition, then Forth cannot find the verb you just defined unless you enter *SMUDGE* after the definition is entered. This is because the *;* verb which is used to end a definition reverses the *SMUDGE* that *:* did to hide the verb from *-FIND*.

To avoid the accidental occurrence of this condition, try the variation in Listing 2. I have defined the verbs - | . and | . , which are used in pairs. The | . verb leaves an 8 on the stack to be checked by the | . verb using the verb *?PAIRS*.

This is the standard method of providing compiler security that is used in such verb pairs as *IF* . . . *THEN, BEGIN* . . . *UNTIL*, etc., and is a good technique to use when defining immediate verbs that are used in pairs. This method will give an error if the second of the pair is used without the first of the pair, if paired verbs are mixed incorrectly, or if the second verb of a pair is not used before the definition is ended, so quite a bit of checking is done in very little code.

A method of recursion was shown in *Forth Dimensions*, vol. 4, no. 2 (published by the Forth Interest Group, P.O. Box 1105, San Carlos, Calif. 94070) that is worth reviewing. The technique was used in a Forth decompiler. *MYSELF* uses the standard fig-Forth verb *LATEST*, which leaves the NFA of the last word in the dictionary and is unaffected by *SMUDGE*. The NFA is then changed to a PFA, then to a CFA, which is compiled. Since the last word in the dictionary is the verb being defined, the verb then calls *itself*.

Recursion as a programming technique can be very useful but do not be led to believe that a recursive routine is always superior to an iterative one. A.K. Dewd-

```
( CRASH - determine size of return stack
 by crashing FORTH )

: MYSELF LATEST PFA CFA , ; IMMEDIATE  ( call myself )

: RECURSE 1+ DUP . MYSELF ;

: CRASH 1 RECURSE ;
```

Listing 1.

```
( FIBONACCI using BAR with compiler security )

: |.  SMUDGE 8 ; IMMEDIATE

: .|  8 ?PAIRS SMUDGE ; IMMEDIATE

: ONACCI                                ( n1 --- n2 )
  DUP 1 = IF DROP 0
          ELSE DUP 2 = IF DROP 1
                       ELSE DUP  1 - |. ONACCI .|
                            SWAP 2 - |. ONACCI .| +
                       THEN
          THEN ;

 : FIB ONACCI U. ;                       ( n --- )
```

Listing 2.

ney pointed out in the Computer Recreations column in *Scientific American* several months ago that the Towers of Hanoi problem can be programmed as a recursive algorithm. (The Towers of Hanoi has three pegs. On one of these pegs is *n* disks of graduated sizes, the larger on bottom. The object of the puzzle is to move all of the disks from the original peg to any other, moving one disk at a time, and never placing a larger disk on a smaller one.)

Indeed, it appears at first to be a prime example of a recursion. To move all the disks, all you have to do is move disk *n* (the largest disk). To move disk *n*, all you have to do is move disk *n*−*1*, etc.

Dewdney points out that an iterative routine is not only easier, it is more efficient. In the routine, the pegs are viewed as being in a triangle. Here's the algorithm:

1. Move the smallest disk clockwise
2. Move any other disk (that is a legal move)
3. Repeat until all of the disks are on another peg.

Forth is a very interesting language that has a lot of power. Serious professionals would be wise to learn more about it.

Bill Hall
Burleson, Texas

Author Jean-Pierre Schachter responds: Bill Hall is quite right about the return stack, and I thoroughly agree that it is a wise precaution to find its limits ahead of time.

As far as the point about odd numbers of BAR *implementations is concerned, I'd first stress that such a case would clearly be one of error—there would be no legitimate use in which the* BAR *would appear in an odd number. Should one wish to build in error checking, Mr. Hall's device seems perfect for the task. However, I should add that the possibility of accidently omitting a* BAR *might recommend simply using* MYSELF *instead of bothering to modify the* BAR *at all.*

Both recursion words work; the fact that the BAR *must be used in two places rather than one might be a reason for preferring* MYSELF*. On the other hand, I'm inclined to prefer the* BAR *for being a conceptually simpler piece of machinery. I do rather like Mr. Hall's error-checking device.*

A PROTOTYPE FOR YOUR PROTOTYPES

Methods

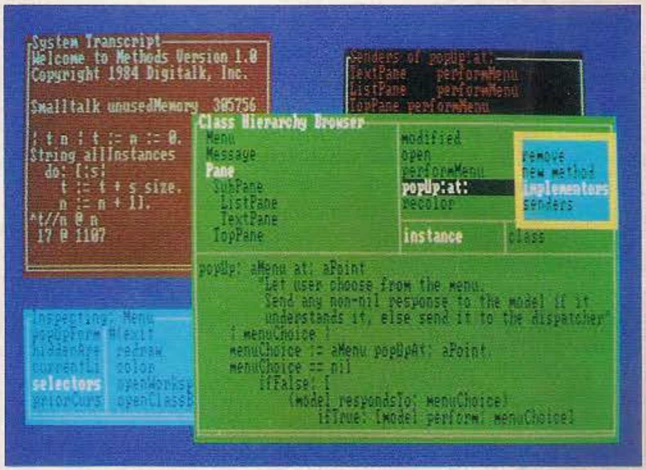
METHODS BY DIGITALK. A SMALLTALK PROGRAM DEVELOPMENT ENVIRONMENT FOR THE IBM PC. AN OBJECT-ORIENTED SPEED DEMON. AN EXCITING WAY TO PROGRAM.

Think about your problem. Browse the Methods Smalltalk source code. Select some building blocks like pop-up menus, windows, text-editors and dictionaries. Put them together. Try it. Oops, Methods tells you something is missing. No problem. Continue thinking, changing and trying. Now you're prototyping! Try out new ideas. Redefine your problem. And you don't have to throw this prototype away. Refine it until you like the finished product.

Methods by Digitalk. A new way to develop software for the PC. Use it for windowing, simulation and artificial intelligence applications. Use it by prototyping.

Methods is Smalltalk-80™ language compatible. It includes its own Smalltalk source code. You can extend it in Smalltalk and assembly language.

Methods operates on IBM PCs with 512K bytes RAM using MS-DOS or PC-DOS. Color and monochrome monitors are supported. No mouse is required. A Smalltalk language manual and an environment guide are included.



```
System Transcript
Welcome to Methods Version 1.0
Copyright 1984 Digitalk, Inc.

Smalltalk unusedMemory 385756

| t n | t := n := 0.
String allInstances
  do: [:s|
    t := t + s size.
    n := n + 1].
^t//n @ n
 17 @ 1107
```

```
Senders of popUp:at:
TextPane    performMenu
ListPane    performMenu
TopPane performMenu
```

```
popUp: aMenu at: aPoint
        "Let user choose from the menu.
        Send any non-nil response to the model if it
        understands it, else send it to the dispatcher"
    | menuChoice |
    menuChoice := aMenu popUpAt: aPoint.
    menuChoice == nil
        ifFalse: [
            (model respondsTo: menuChoice)
                ifTrue: [model perform: menuChoice]
```

Available from Digitalk for $250. Outside U.S. add $15.00 for shipping and handling. California residents add 6% sales tax. Visa and MasterCard accepted. Educational and Dealer Discounts Available.

DIGITALK, INC.
5200 West Century Boulevard
Los Angeles, California 90045
(213) 645-1082

Smalltalk-80™ is a trademark of Xerox Corporation. IBM is a registered trademark of International Business Machines Corporation. MS is a trademark of Microsoft Corporation.

CIRCLE 45 ON READER SERVICE CARD

SUBSCRIBE TODAY AND SAVE MORE!

YES! Please start my subscription to COMPUTER LANGUAGE for great savings off the single-copy price! Enter my subscription for the term marked below.

☐ 1 year (12 issues) for only $24.95—a 30% savings
☐ 2 year (24 issues) for $39.95—save 44%
☐ 3 year (36 issues) for $49.95—save 53%

☐ **Bill me later.** ☐ **My check is enclosed.**

Name ______

Company ______

Address ______

City, State, Zip ______

Please allow 4–6 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canadian orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail. BI85

SUBSCRIBE TODAY AND SAVE MORE!

YES! Please start my subscription to COMPUTER LANGUAGE for great savings off the single-copy price! Enter my subscription for the term marked below.

☐ 1 year (12 issues) for only $24.95—a 30% savings
☐ 2 year (24 issues) for $39.95—save 44%
☐ 3 year (36 issues) for $49.95—save 53%

☐ **Bill me later.** ☐ **My check is enclosed.**

Name ______

Company ______

Address ______

City, State, Zip ______

Please allow 4–6 weeks for delivery of first issue. Foreign orders must be prepaid in U.S. funds. Canadian orders $30.95 per year. Outside the U.S., $36.95/year for surface mail or $54.95/year for airmail. BI85

C Seminar/Workshop

Sept. 16–18
Cambridge, Mass

Creators of COMPUTER LANGUAGE Sponsor the C Expert Forum

Never before have so many leaders in the C programming field gathered for one event. The C Seminar/Workshop will be an exciting forum on the latest technical innovations and C language developments. Best of all, you'll experience a practical, hands-on approach in small workshop sessions.

Here are just a few of the speakers and workshop leaders:

Jim Brodie
Ron Cain
Scott Gimpel
Leo Kenen
P.J. Plauger
Larry Rosler
Tom Plum
Jack Purdum
Heinz Lycklama
Bob Varney
Robert Ward
Leor Zolman

☐ **Please enroll me in the C Seminar**, registration $695.
☐ **Check enclosed.** (Make check payable to: C.L. Publications Inc.)
☐ **Bill my company.**

Name ______

Company ______

Address ______

City, State, Zip ______

Phone ______

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115

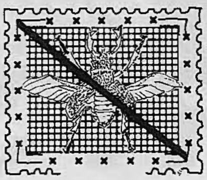
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 22481 SAN FRANCISCO, CA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

2443 FILLMORE STREET • SUITE 346
SAN FRANCISCO, CA 94115



[redacted]

Modem7's evolution

Dear Editor:

I particularly enjoyed Craig LaGrow's "ComputerVisions—On-line with Ward Christensen" (*COMPUTER LANGUAGE*, June 1985, pp. 71-78). It's always nice to read about one's personal heroes. However, I was disappointed to learn that Christensen seems to have no interest in the latest versions of the modem programs originated by himself. That's too bad, because all who use the current versions of Modem7 (now called MDM7), XMODEM, and BYE credit him as the father upon whose work the newer stuff is based.

It's true—Christensen probably would not recognize any of the recent releases of the MDM7 program series as related to his original Modem7. Others, and most particularly Irv Hoff, have added so many features that the software now competes directly with commercial products costing many dollars. The big difference, of course, is that Hoff has picked up the role of torchbearer and kept his MDM7 series in the public domain—100% free (another Christensen tradition). For the price of a phone call, anyone can obtain copies of the MDM740, XMODEM-106, and the BYE-326 libraries from one of the hundreds of RCP/M systems around the world.

Your readers may be interested in knowing that Irv Hoff and systems operator Wayne Masters have assumed clearinghouse responsibilities for the Modem7 descendants. Current releases are available from their POTPOURRI RCP/M in San Jose, Calif., at (408) 378-7474.

Masters reports that later this summer new versions of MDM7 and XMODEM will be released which complement the greater speeds of 1200- and 2400-baud modems. The newest feature will be a 1K block transfer protocol using 16-bit CRC verification, which will enhance data throughput to approximate 95% efficiency. The price will remain the same—free, plus the cost of a phone call!

From this point forward, if not before, I hope we'll all be sure and remember to include their [Hoff and Masters] names when it's time to give credit where it's due. Surely they deserve no less for their yeomanlike labors expended on behalf of the backbone of personal computer modem communications.

Bond Shands
San Francisco, Calif.

[redacted]

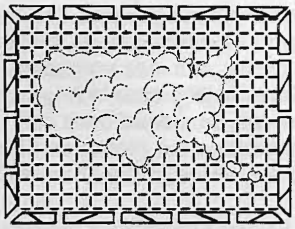
Not-so-simple sort

Dear Editor:

M.B. Clausing pointed out several problems with his simple sort algorithm (*COMPUTER LANGUAGE*, June 1985, p. 9). Unfortunately, the most important flaw was missed: it does not work. It can fail for sets of data with as few as four elements. In particular, it incorrectly sorts each of the following three ordered quadruples (and uncountably more data sets of greater size): (2, 3, 4, 1), (3, 4, 2, 1), and (4, 3, 1, 2).

The algorithm can be fixed in a variety of ways; however, as Clausing stated, "attempts to improve it mutate it into more familiar algorithms."

Daniel L. Stock and
Randall L. Brukardt
Madison, Wis.



[redacted]

COBOL improvements

Dear Editor:

In "Back to the Drawing Board—Electronic soapboxes" (*COMPUTER LANGUAGE*, Dec. 1984, pp. 15-18), Greg Law complains: "COBOL, the only program to require 300 lines of initialization per line of code . . . Talk about hassles." COBOL-85, available already from DEC and some other vendors, contains three enhancements to reduce this problem, specifically:

- A *VALUE* clause can now be written in an entry with a table element; this initializes all elements to some single value, avoiding the need for a long, separate entry.
- The new *INITIALIZE* verb can initialize an item to a default of zeros or spaces, depending on its category, or to some specified value. If a group item is specified, its elementary items are initialized individually, by category. Multi-item operands can also be specified.
- The new *INITIAL* clause of the *PROGRAM-ID* paragraph implicitly reinitializes all data items in a subprogram to their *VALUE*s whenever the subprogram is called.

COBOL-9x, now being put together, contains two further features:

- The *VALUES ARE* clause can now be used in a table entry; its multiple operands initialize individual elements to different values. This eliminates the need to have separate entries for the *OCCURS* clause and the literal, the need to count out inter-element blanks in the literal, and the unreadability that occurs when such a literal wraps around a line. Most usefully, it enables data division initialization of elementary items of different categories that are subsidiary to a group item with an *OCCURS* clause.
- The *INITIALIZE* verb adds the *TO VALUE* phrase, enabling the reinitialization of any data item, even a complicated table, to its initial *VALUE*(s).

These additions will turn COBOL from a chump to a champ at initialization. If any vendors are listening who would like to make COBOL more lovable, I suggest they add the last two items as extensions to their 1985 compilers. (They might charge a bit extra for them, I suppose.)

Roger Knights
Seattle, Wash.

[redacted]

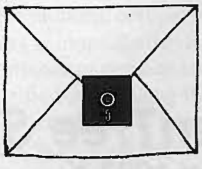
Unsolicited—honest!

Dear Editor:

Gimme a break, willya? Ever since a colleague strolled in to my office with a comparison of 21 C compilers (*COMPUTER LANGUAGE*, Feb. 1985, pp. 73-102), my productivity has plummeted. Stacks of *EDN, Computer Design, EE Times, Digital Design, Electronic Design, MiniMicro Review, Computer World*, etc., are occluding the view of my desk. What right have you folks got to publish a magazine where more than 75% of the material is not only interesting but pertinent to me as a software professional? When am I supposed to find the time to do my work, much less browse through the pulps? And now that I have just received all the back issues, it is apparent to me that the situation will only worsen in the near term!

No periodical I have read (from *CACM* to *Dr. Dobbs*) seems to address the exact niche you are carving, which lo and behold is the niche I must be in. To make matters worse, I have spent more than a month's pay in the last two days buying software and tools I saw advertised in *COMPUTER LANGUAGE*. So I repeat, GIMME A BREAK, WILLYA? If if keeps up, I'll be both broke and buried in old pulps. Thanks for the forum.

Chris Pinkard
Austin, Texas



Illustrations: Anne Doering

At last! - Fast, On-screen

FLOWCHARTS

And Organization Charts

Finally! An on-screen flowchart processor that knows about flowcharts and organization charts - not just another "screen draw" program that makes you do most of the work.

Interactive EasyFlow is a powerful full-screen graphics program dedicated to flowcharts and organization charts. This program allows you to easily compose charts on the screen.

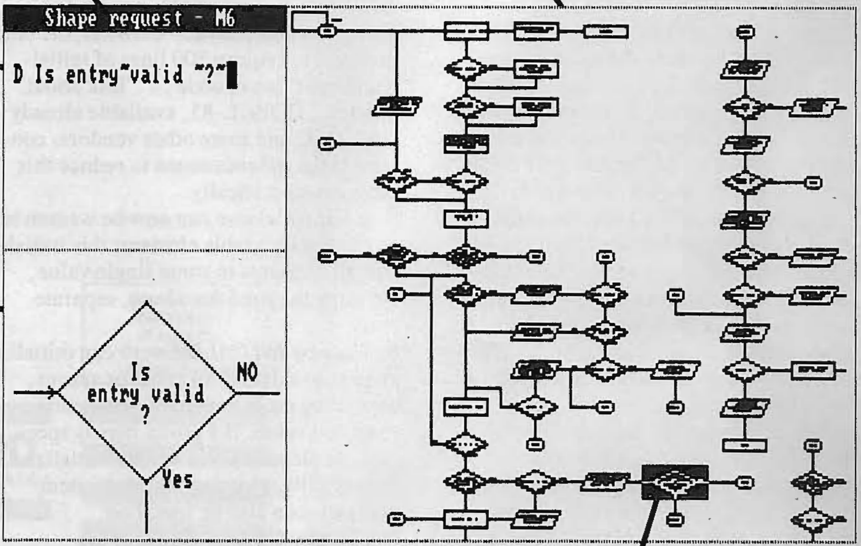
Features: ● Text is automatically centered, character by character, within shapes as you type it. ● Lines are created by specifying the starting and ending points - the program automatically generates the route. ● Powerful editing facilities allow shapes and even entire rows and columns of shapes to be inserted or deleted; lines are automatically re-routed as necessary. ● Large chart size (up to 16 shapes wide by 16 shapes high) allows very large flowcharts and organization charts to be handled with ease. ● Charts can be larger than the screen - the window into the chart scrolls both horizontally and vertically as necessary. ● Flexible printer interface allows it to work with all printers, not just dot matrix printers. Wide charts can be printed in strips. ● Eighteen standard flowcharting shapes included. ● User defined shapes can easily be added. ● The manual is extensive and includes many examples. ● On-line "help" facility provides immediate assistance at any time. ● Any number of titles can be placed on a chart. ● Commentary text blocks can be placed anywhere in the chart. ● Plus many more features than we can mention here.

Requires an MSDOS / PCDOS machine equipped with an IBM compatible graphics adapter and at least 256K of memory. Runs under DOS 1; contains full support for DOS 2.

Only $149.95 + $2.00 S&H (USA/Canada), $10.00 (foreign).

Payment by check, M.O., VISA, COD or Company PO.

The sample screen display shown below is typical of what you see while editing a chart. Other screen displays are provided for entering titles, changing options, getting "help" and so on.



STATUS BAR (not to be confused with a wet bar) tells you what Interactive EasyFlow is doing at all times.

CHART WINDOW gives an overview of your chart; this example shows the "normal" view. "Close-up" view shows a smaller part of the chart in more detail. "Wide-angle" view shows a larger part of the chart at reduced size.

TEXT/MESSAGE WINDOW used to enter user text and to display messages from Interactive EasyFlow.

CURRENT SHAPE WINDOW - shows the content of the current flowchart shape in complete detail.

SHAPE CURSOR shows where you are in the chart. Cursor keys move it around; chart window scrolls if you run off the edge of the window.

HavenTree Software Limited

P.O. Box 1093-K
Thousand Island Park, NY 13692
(613) 542-7270 Ext 46

CIRCLE 67 ON READER SERVICE CARD

CROSS X THOUGHTS

Virtual memory data simulation

By Namir Clement Shammas

Virtual memory is a memory management technique that has been employed since the early days of mainframe computers with 4K to 16K memory. VM allows programs and data to exceed the actual or real hardware addresses. This is possible through the use of mass storage (both floppy disks and hard disks) as an extension of real memory and the efficient swapping of required memory sections.

One may recall the Apple-UCSD operating system implementing a program-swapping technique to enable large UCSD Pascal programs to run on microcomputers. UCSD BASIC implements virtual variables, which allows very large matrices and arrays to be declared as virtual variables and to be used like any other BASIC memory-resident variable. In this system, the BASIC interpreter is solely responsible for swapping data between memory and disk. This month's column will cover the simulation of these variables.

Before we continue, I want to point out that it looks like MS-DOS 4.0 will include virtual memory management. Regarding micros, it appears the subject not only is behind us but also ahead of us. The simulation of virtual data structures uses techniques similar to those in operating systems.

Let's begin with a discussion of the simulation of virtual one-dimensional arrays.

Basically, we assign the array a virtual size that is actually too big to be contained in RAM but can be stored in mass storage devices. Part of the array can be contained in RAM at any time and portions can be swapped systemically between RAM and mass storage as needed. Handling virtual memory requires speed; without it, the application program suffers from intolerable slowness.

Since we are dealing with buffered I/O, we must transfer pages instead of individual data items. This takes care of the inefficiency that comes from handling too little data, but having to work with large pages also causes time loss. Suppose a 20K page is used and we need to update only 5K of modified information to mass storage. We would be wasting time rewriting 15K of unaltered data!

Fortunately, the strategy of divide and conquer comes to the rescue once again. Instead of having only one big page resident in RAM, we employ a number of smaller, equal-sized pages. This enables us to localize any altered elements that need to be swapped with mass storage. In other words, we obtain higher resolution for data I/O. The chosen page size acts as a yardstick in counting and mapping the total number of pages representing the virtual array.

We also need to maintain a small RAM-resident table to indicate the pages that are in RAM. This table can be searched to see if the sought page is already in RAM, thus avoiding a blind and time-consuming recall from mass storage.

Another simple measure we can adopt to decrease the I/O time is assigning each page a Boolean flag or bit, called the dirty bit. When a page is recalled into memory, its dirty bit is set to zero, declaring it clean. As long as data is read only from a page, its dirty bit remains zero. But once data is written to a page, the dirty bit is set to one and declared dirty. When it comes time for a page to be swapped back to the disk to make way for another page, the dirty bit is used to determine whether the page actually needs to be written back. The small overhead of bits saves us from performing unnecessary output to mass storage.

The PPL code in Listing 1 is a function that takes the virtual array index and returns the index of a RAM-resident array. It is important to note that while a large single array is used to store all the RAM pages, we swap page-size bytes of the array at a time. The RAM array does not contain the pages in accordance with their virtual sequence. The map table must be used to determine exactly where a sought element is located. The function *LOC* takes the virtual array index and calculates the index of the virtual page.

Next we search the map table entries to see if the sought page is already in memory. If the search is successful, we assign the entry number value to the sought page number, transforming the virtual page number into a RAM page number. On the other hand, when the search fails we swap a least-desirable RAM page.

The page read is entered into the map table and the entry number is used as the RAM page number. The exact location is obtained as shown by the return statement in *LOC*. The procedure *VM_Assign* relies on the return statement in *LOC*, performs the actual array assignment, and sets the dirty bit to one.

The matrix is another popular data structure used in virtual memory. Since it is a two-dimensional array we can unwrap it into one big array and use the *VM_Assign* approach. A matrix can be transformed into a series of chained rows or columns depending on how convenient it is for the application.

What about when it is more efficient to access the virtual matrix in pages containing submatrices? This resembles the way spreadsheets are viewed on microcomputers: you normally see a portion of the entire table. We will call this the checkerboard method, where the entire matrix is divided into similar and smaller structures.

The virtual matrix is divided into equal-sized pages, each containing the same number of rows and columns. To achieve a perfect fit, the virtual matrix may be slightly larger than our arbitrarily sized matrix. The partitioned matrix yields a matrix of pages. The latter pages are unwrapped into continuous arrays.

We can adopt one of two page-numbering systems. The first, double indexing, specifies each page by a row and column coordinate. The second, single indexing, systemically numbers pages row-wise or column-wise. The page numbers are used as pointers and stored in the map table.

Since it is faster to search for a page using a single numeric comparison, single page indexing is more desirable. But the maximum number of rows or columns must be specified when using this method, allowing the virtual matrix to grow only in one direction. With double indexing, the matrix can grow in either direction.

Listing 2 shows a function that accepts

the row and column indices of a virtual matrix and returns the exact index location of the RAM-resident array containing the submatrix pages. A single page index is calculated and used in the map table search.

The method shown in Listing 2 is similar to virtual one-dimensional arrays. In this method we first attempt to locate the sought page in memory. If this fails, we select a page to swap and then bring in the sought page. With the sought page known, we close in on the RAM array index by calculating the row and column index of the submatrix. The *return* statement shows the expression used to map our submatrices into a large one-dimensional array index.

Let's now focus our attention on the page replacement algorithms. In Listing 1 and Listing 2 we use a *Selected_Page()* function that returns the page number of the best page to replace. This plays an important role in the speed of the simulated virtual system.

In considering the various algorithms to use, I am assuming that we are integrating the concept of the dirty bit to cover some potential deficiencies. The algorithms to be considered follow.

Random page replacement (RPR). This method uses a very low overhead in selecting the page to be randomly replaced. Thus, all RAM-resident pages have an equal probability of being replaced. But this method doesn't really pay off in the long run because the savings in calculation time is offset by the inefficient and time-consuming data I/O.

First in-first out (FIFO). When getting ready to swap, this algorithm looks at page precedence and selects the page that has been in memory the longest. We use a stack to keep track of precedence. When the sought page is already in memory we manipulate the stack to reposition the page number at the top of the stack. Otherwise we roll the stack so that the oldest element ends up on top, ready to be replaced by the incoming page number.

Optimal replacement. Theoretically, this method is the best. It is based on knowing in advance the pattern of recalling pages. Using this information, we replace the RAM page that won't be used for the longest time. Easier said than done! While this method is difficult to implement, other methods try to approach this same concept.

Least-recently-used (LRU) page replacement. This method tries to approximate the optimal replacement method by attempting to predict the near future based on the recent past. Thus we replace pages that have not been used for the longest period of time. While this method seems to present a very good solution, special attention must be paid to how it is implemented.

There are two forms of implementation. The first alternative is to time-stamp page number entries in the map table as they are used. To select the page to be replaced, we search through the map table for the oldest entry or the entry with least time. If the sought page is already in RAM, we restamp it to update its chronological usage. This method requires that the *Table_Type* record include a time field. The second alternative is to put the recalled page numbers in a stack, which gives an indication of how recently a page was read.

Least-frequently-used (LFU) page replacement. This method is similar to LRU page replacement, but instead of time-stamping page entries, a frequency counter field is employed. As RAM resident pages are referenced their counters are incremented. Pages are replaced by searching the map table for the page with the lowest frequency of use. But this can have an adverse effect on new pages with low usage frequency. They may be replaced only to be read within two page recalls. Only sequentially referenced pages will manage to build a good frequency count fast enough.

Modified, least-frequently-used (MLFU) page replacement. This method is a modification of LRU and LFU page replacement. By employing a time-stamp, it overcomes the tendency of LFU to cause recently read pages to be replaced more likely than other pages. This gives the candidate-for-replacement-page a second chance to remain in RAM because it was recently read and may come into use in the near future.

Listing 3 shows the PPL code for the MLFU method implemented by a version of the *Selected_Page* function. Note the use of two consecutive loops. The search for least recently used pages is carried out in the first loop. The second loop searches for a candidate replacement page according to frequency of usage and residence time in memory. It is important to note that the *Critical_Time* identifier must be systematically updated in other program segments to maintain meaningful results.

Assigned static or dynamic page priority (ASDPP). In some cases, certain data subsets of an array or matrix are more frequently used than others. The user can assign priorities to the virtual pages. These priorities can be page/priority lookup tables or functions that return the priority value if its values are assigned over ranges of indices. The first option is faster but is memory-consuming, while the second is slower but is memory-conserving.

Static priorities are assigned at the beginning of the program and remain fixed. Dynamic priorities are altered during program execution, reflecting the fact that different data areas have become "hot spots." To replace a RAM page, we search the map table and locate the RAM page with the least priority.

Not-used-recently (NUR) page replacement. This algorithm is based on the concept that a page not recently used is not likely to be used in the near future. This involves a small overhead of two Boolean flags or bits. The first is the reference bit used to signal whether a RAM page was referenced at least once after it was brought from mass storage. The second bit is the dirty bit.

The selection is simple and consists of three steps. First, search the map table for an unreferenced page. If this is not successful, search the table for an unmodified page. If this fails, select the preassigned default page. I should point out that the dirty bit should be set by modified versions of the procedures *VM_Assign* and *VM_MAT_Assign.*

Implementing virtual memory is a double-edged sword. A bad choice of algorithms can make the system run very slowly. Allowing the program to run unattended overnight relieves the pressure of pushing the speed to the limit. It does not matter if the program ends execution at 3 a.m. or 5 a.m., there is no one at the office anyway! To gain speed on microcomputers, one can use virtual memory with a RAM electronic disk. This method is useful when the language interpreter/compiler does not allow the use of the entire memory.

Next month I will discuss high-resolution graphics and the mathematics behind two- and three-dimensional shapes and figures. I will also talk about mapping real-world coordinates to hardware screen coordinates, including windows and logarithmic scales. The algorithms for plotting three-dimensional surfaces with hidden areas also will be discussed.

```
PPL routines to recall and store elements of a one-dimensional
virtual array

-- Constants are:

-- Table_Size            Maximum number of pages in RAM.
-- Page_Size             Number of array elements in a page.
-- Array_base_Address    Lower array limit in a page.
-- VM_base_address       Starting VM address

-- External identifiers are:

-- Table_Type = RECORD RAM_Page, Dirty_bit : integer; END
-- Table : ARRAY[1..Table_Size] of Table_Type
-- SOUGHT_PAGE : integer
-- RAM_Data : ARRAY[1..PageSize] of Any type

-- File VM_File is assumed already opened

FUNCTION LOC(VM_Index : integer) return integer

BEGIN
    SOUGHT_PAGE = VM_Index div Page_Size
    -- Loop to search for page among RAM resident pages.
    INITIALIZE: Found = FALSE
    LOOP <Table_Lookup>
    BEGIN For i = 1 TO Table_Size
        IF SOUGHT_PAGE = Table[i].RAM_Page
        THEN Found = True
             EXIT <Table_Lookup>
        END IF
    END LOOP <Table_Lookup>
    TERMINATE: None

    IF NOT Found
    THEN
        Page_Swapped = Selected_Page() -- Function to select RAM resident
                                          Page for swapping
        IF Table[Page_Swapped].Dirty_bit THEN SAVE(Page_Swapped) END IF
        LOAD(SOUGHT_PAGE,Page_Swapped)
        Table[Page_Swapped].RAM_Page = SOUGHT_PAGE
        Table[Page_Swapped].Dirty_bit = 0
        SOUGHT_PAGE = Page_Swapped
    ELSE
        Page_Swapped = Selected_Page() -- Dummy call used to update
                                       -- page replacement history
        SOUGHT_PAGE = i
    END IF
    return (SOUGHT_PAGE * Page_Size + VM_Index mod Page_Size
             + Array_base_address)

END LOC
```

Listing 1. *(Continued on following page)*

PERFORMANCE PACKAGE

Blaise Computing Inc. introduces the PERFORMANCE PACKAGE™ for Turbo Pascal programmers.

TURBO PASCAL

Turbo ASYNCH™

With Turbo ASYNCH, you can be in constant touch with the world without ever leaving the console. Rapid transit at its best. Turbo ASYNCH is designed to let you incorporate asynchronous communication capabilities into your Turbo Pascal application programs, and it will drive any asynchronous device via the RS232 ports, like printers, plotters, modems or even other computers. Turbo ASYNCH is fast, accurate and lives up to its specs. Features include…

◆ Initialization of the COM ports allowing you to set all transmission options. ◆ Interrupt processing. ◆ Data transfer between circular queues and communications ports. ◆ Simultaneous buffered input and output to both COM ports. ◆ Transmission speeds up to 9600 Baud. ◆ Input and output queues as large as you wish. ◆ XON/XOFF protocol.

The underlying functions of Turbo ASYNCH are carefully crafted in assembler for efficiency, and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The interface to the assembler routines is written in Turbo Pascal.

The Turbo Pascal PERFORMANCE PACKAGE™ is for the serious Turbo Pascal programmer who wants quality tools to develop applications. Every system comes with a comprehensive User Reference Manual, all source code and useful sample programs. They require an IBM PC or compatible, utilizing MS-DOS version 2.0 or later. There are no royalties for incorporating PERFORMANCE PACKAGE functions into your applications.

Turbo POWER TOOLS and Turbo ASYNCH sell for $99.95 each, and they may be ordered directly from **Blaise Computing, Inc. TO ORDER, call (415) 540-5441**

NEW!

Turbo POWER TOOLS™

Turbo POWER TOOLS is a sleek new series of procedures designed specifically to complement Turbo Pascal on IBM and compatible computers. Every component in Turbo POWER TOOLS is precision engineered to give you fluid and responsive handling, with all the options you need packed into its clean lines. High performance and full instrumentation, including…

◆ Extensive string handling to complement the powerful Turbo Pascal functions. ◆ Screen support and window management, giving you fast direct access to the screen without using BIOS calls. ◆ Access to BIOS and DOS services, including DOS 3.0 and the IBM AT. ◆ Full program control by allowing you to execute any other program from within your Turbo Pascal application. ◆ Interrupt service routines written entirely in Turbo Pascal. Assembly code is not required even to service hardware interrupts like the keyboard or clock.

Using Turbo POWER TOOLS, you can now "filter" the keyboard or even DOS, and create your own "sidekickable" applications.

YES, send me the Best for the Best! Enclosed is ________ for
☐ Turbo ASYNCH ☐ Turbo POWER TOOLS. (CA residents add
6½% Sales Tax. All orders add $6.00 for shipping.)

Name: ____________ **Phone:** ____________
Shipping Address: ____________ **Zip:** ____________
City: ____________ **State:** ____________
VISA or MC #: ____________ **Exp. Date:** ____________

Turbo Pascal is a trademark of Borland International. Turbo POWER TOOLS, Turbo ASYNCH and PERFORMANCE PACKAGE are trademarks of Blaise Computing Inc. IBM is a registered trademark of International Business Machines Corporation. MS-DOS is a trademark of Microsoft Corporation.

BLAISE COMPUTING INC.

BLAISE *watch us!*

◆ 2034 BLAKE STREET
◆ BERKELEY, CA 94704
◆ (415) 540-5441

CIRCLE 1 ON READER SERVICE CARD

```
PROCEDURE VM_Assign(VM_Index : integer; X : Your_data)

BEGIN
    RAM_Data[LOC(VM_Index)] = X
    Table[SOUGHT_PAGE].Dirty_bit = 1
END VM_Assign


PROCEDURE SAVE(Page_Number : integer)

BEGIN
 INITIALIZE: m = Page_Number - 1) * Page_Size + Array_base_address
             p = (Table[Page_Number].RAM_Page - 1) * PageSize +
                 VM_base_address
     LOOP
     BEGIN  For n = 1 to Page_Size
          WRITE VM_File, (p + n),RAM_Data[m + n]
     END LOOP
END SAVE
PROCEDURE LOAD(VM_Page, RAM_Page_Num : integer)

BEGIN
 INITIALIZE: m = RAM_Page_Num - 1) * Page_Size + Array_base_address
             p = (VM_Page - 1) * PageSize  + VM_base_address
     LOOP
     BEGIN  For n = 1 to Page_Size
          READ VM_File, (p + n),RAM_Data[m + n]
     END LOOP
END SAVE
```

Listing 1. *(Continued from preceding page)*

```
PPL routines to recall and store elements of a two-dimensional
virtual array, using the checkerboard method

-- Constants are:

-- Table_Size : Maximum number of pages in RAM.
-- Page_Col_Size  : Number of columns in a page.
-- Page_Row_Size  : Number of rows in a page
-- ROWS : Number of rows

-- External identifiers are:

-- Table_Type = RECORD RAM_Page, Dirty_bit : integer; END
-- Table : ARRAY[1..Table_Size] of Table_Type
-- SOUGHT_PAGE : integer
```

Listing 2. *(Continued on a following page)*

```
-- RAM_Data :
    ARRAY[1..(Table_Size * Page_Col_Size * Page_Row_Size)] of Any type

-- Procedures SAVE and LOAD are similar to those in listing (1),
the difference being is that they incorporate wrapping and
unwrapping arrays to matrices and vice versa, respectively.


FUNCTION MATLOC(VM_Row_Index, VM_Col_Index : integer) return integer

BEGIN
    SOUGHT_PAGE_Row = VM_Row_Index div Page_Row_Size + 1
    SOUGHT_PAGE_Col = VM_Col_Index div Page_Col_Size + 1
    SOUGHT_PAGE = SOUGHT_PAGE_Row  + (SOUGHT_PAGE_Col - 1) * ROWS
    -- Loop to search for page among RAM resident pages.
    INITIALIZE: Found = FALSE
    LOOP <Table_Lookup>
    BEGIN For i = 1 TO Table_Size
        IF SOUGHT_PAGE_Row = Table[i].RAM_Page
        THEN Found = True
             EXIT <Table_Lookup>
        END IF
    END LOOP <Table_Lookup>
    TERMINATE: None

    IF NOT Found
    THEN
        Page_Swapped = Selected_Page() -- Function to select RAM resident
                                          Page for swapping
        IF Table[Page_Swapped].Dirty_bit THEN SAVE_MAT(Page_Swapped) END IF
        LOAD_MAT(SOUGHT_PAGE,Page_Swapped)
        Table[Page_Swapped].RAM_Page = SOUGHT_PAGE
        Table[Page_Swapped].Dirty_bit = 0
        SOUGHT_PAGE = Page_Swapped
    ELSE
        Page_Swapped = Selected_Page() -- Dummy call used to update
                                       -- page replacement history
        SOUGHT_PAGE = i


    END IF
    Page_Row = VM_Row_Index mod Page_Row_Size + 1
    Page_Col = VM_Col_Index mod Page_Row_Size + 1
    Mat_Size = Page_Row_Size * Page_Col_Size
    return {(SOUGHT_PAGE - 1) * Mat_Size + (Page_Col - 1) * Page_Row_Size
             + Page_Row}

END MATLOC


PROCEDURE VM_MAT_Assign(VM_Index : integer; X : Your_data)

BEGIN
    RAM_Data[LOC(VM_Row_Index,VM_Col_Index)] = X
    Table[SOUGHT_PAGE].Dirty_bit = 1
END VM_MAT_Assign
```

Listing 2. *(Continued from a preceding page)*

Slash Programming Time in Half!

With FirsTime™

- Fast program entry through single keystroke statement generators.
- Fast editing through syntax oriented cursor movements.
- Dramatically reduced debugging time through immediate syntax checking.
- Fast development through unique programmer oriented features.
- Automatic program formatter.

FirsTime is a true syntax directed editor.

FirsTime ensures the integrity of your programs by performing all editing tasks like moves, inserts and deletes along the syntactic elements of a program. For example, when you move an IF statement, FirsTime will move the corresponding THEN and ELSE clauses with it.

Even FirsTime's cursor movements are by syntax elements instead of characters. The cursor automatically skips over blank spaces and required keywords and goes directly to the next editable position.

FirsTime is a Syntax Checker

FirsTime checks the syntax of your program statements, and also:

- Semantics like undefined variables and mismatched statement types.
- The contents of include files and macro expansions.
- Statements for errors as they are entered and warns you immediately.

FirsTime is a Program Formatter

FirsTime automatically indents statements as they are entered, saving you from having to track indentation levels and count spaces.

FirsTime has Unique Features

No other editor offer these features:

The *Zoom command* gives you a top down view of your program logic.

The *View command* displays the contents of include files and macro expansions. This is valuable to sophisticated programmers writing complex code or to those updating unfamiliar programs.

FirsTime's *Transform command* lets you change a statement to another similar one with just two keystrokes. For example, you can instantly transform a FOR statement into a WHILE statement.

The *Move at Same Level command* moves the cursor up or down to the next statement at the same indentation level. This is very useful. For example, you can use it to locate the ELSE clause that corresponds to a given THEN clause or to traverse a program one procedure at a time.

FirsTime is Unparalleled

FirsTime is the most advanced syntax directed editor available. It makes programming faster, easier and more fun.

TO ORDER CALL (201) 741-8188

or write:

Spruce Technology Corporation

189 E. Bergen Place
Red Bank, NJ 07701

In Germany, Austria and Switzerland contact:

Markt & Technik Software Verlag
Munchen, W. Germany
(089) 4613-0

FirsTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation
IBM is a trademark of International Business Machines, Inc. • Turbo Pascal is a trademark of Borland International

CIRCLE 26 ON READER SERVICE CARD

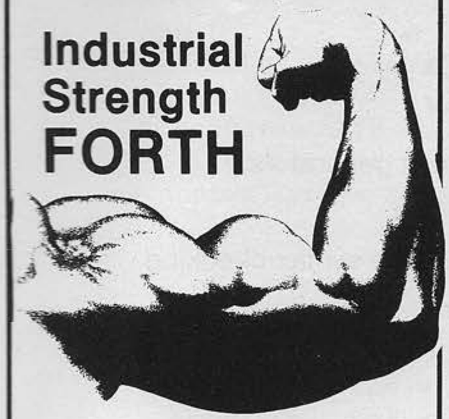
FirsTime for **Turbo Pascal $\$74^{95}$**

FirsTime for **MicroSoft & IBM Pascal $\$245$**

FirsTime for **C $\$295$**

Multiuser/Multitasking
for 8080, Z80, 8086

Industrial Strength FORTH



TaskFORTH™

The First
Professional Quality
Full Feature FORTH
System at a micro price*

LOADS OF TIME SAVING PROFESSIONAL FEATURES:

- ☆ Unlimited number of tasks
- ☆ Multiple thread dictionary, superfast compilation
- ☆ Novice Programmer Protection Package™
- ☆ Diagnostic tools, quick and simple debugging
- ☆ Starting FORTH, FORTH-79, FORTH-83 compatible
- ☆ Screen and serial editor, easy program generation
- ☆ Hierarchical file system with data base management

* Starter package $250. Full package $395. Single user and commercial licenses available.

If you are an experienced FORTH programmer, this is the one you have been waiting for! If you are a beginning FORTH programmer, this will get you started right, and quickly too!

Available on 8 inch disk under CP/M 2.2 or greater also various 5¼" formats and other operating systems

FULLY WARRANTIED, DOCUMENTED AND SUPPORTED

master charge DEALER INQUIRES INVITED VISA

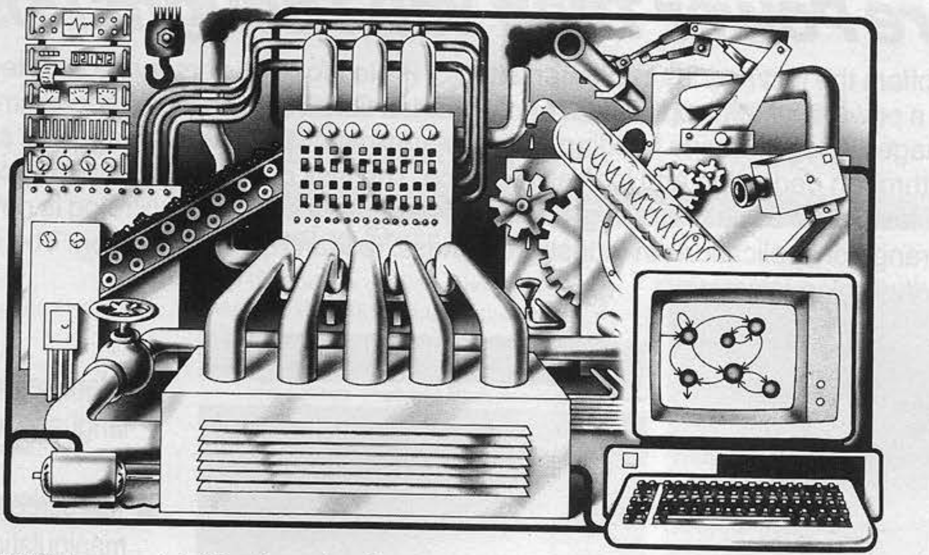
Shaw Laboratories, Ltd.
24301 Southland Drive, #216
Hayward, California 94545
(415) 276-5953

CIRCLE 79 ON READER SERVICE CARD

```
PPL code for modified, least-frequently-used (MLFU) page
replacement function


FUNCTION Selected_Page() return integer

-- Critical_Time is set as a criterion for giving recently read
pages a second chance for staying.  It must be systematically
updated by other program segments.

-- The modified Table_Type is
-- Table_Type = RECORD RAM_Page, Dirty_bit, Frequency : integer;
                       Time : Time_Units END

BEGIN
    IF Found
    THEN   -- when sought page is RAM-resident
        Table[i].Frequency += 1
        return ()
    ELSE
        INITIALIZE: Pick = 1
        LOOP <Start_Up>
        BEGIN For j = 1 to Table_Size
            IF Critical_Time > Table[j].Time
            THEN Least_Frequent = Table[j].Frequency
                 Pick = j
                 EXIT <Start_Up>
            END IF
        END LOOP <Start_Up>
        TERMINATE: None

        INITIALIZE: Least_Frequent = Table[Pick].Frequency
                    Pointer = Pick; j = Pick + 1
        LOOP <Search>
        BEGIN IF j > Table_Size THEN EXIT <Search> END IF
            IF Least_Frequent > Table[j].Frequency AND
               Critical_Time > Table[j].Time
            THEN Least_Frequent = Table[j].Frequency
                 Pointer = j
            ELSE j += 1
            END IF
        END LOOP <Search>
        -- Prepare frequency count for new page
        TERMINATE: Table[Pointer].Frequency = 1
        return( Pointer )
    END IF
END Selected_Page
```

Listing 3.

Csharp Realtime Toolkit



Realtime on MSDOS? Csharp can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: $600

SYSTEMS GUILD

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

CIRCLE 27 ON READER SERVICE CARD

Another in a series of productivity notes on UNIX™ software from UniPress.

Subject: C Cross Compiler for the 8086 Family.

The Lattice C Cross Compiler allows the user to write code on a VAX™ (UNIX or VMS™) or MC68000™ machine for the 8086 family. Lattice C is a timesaving tool that allows a more powerful computer to produce object code for the IBM-PC™. The compiler is regarded as the finest C compiler for the 8086 family and produces the fastest and tightest code.

Features:

- *For your UNIX or VMS Computer.*
- *Use your VAX or other UNIX machine to create standard Intel object code for the 8086 (IBM-PC).*
- *Highly regarded compiler produces fastest and tightest code for the 8086 family.*
- *Full C language and standard library, compatible with UNIX.*
- *Small, medium, compact and large address models available.*
- *Includes compiler, linker, librarian and disassembler.*
- *8087™ floating point support.*
- *MS-DOS™ 2.0 libraries.*
- *Send and Receive communication package optionally available. Price $500.*
- *Optional SSI Intel Style Tools. Package includes linker, locator and assembler and creates executables for debugging on the Intel workstation or for standalone environments. Price $8,550.*

Price:

VAX (UNIX or VMS)	*$5000*
MC68000	*3000*

For more information on these and other UNIX software products, call or write: UniPress Software, Inc., 2025 Lincoln Hwy., Edison, NJ 08817. Telephone: (201) 985-8000. Order Desk: (800) 222-0550 (Outside NJ). Telex: 709418. Japanese Distributor: Softec 0480 (85) 6565. European Distributor: Modulator SA (031) 59 22 22.

OEM terms available.
Mastercard/Visa accepted.

CROSS COMPILER FOR THE 8086™ FAMILY

LATTICE® C CROSS COMPILER

Trademarks of: Lattice: Lattice, Inc.; VAX and VMS: Digital Equipment Corp.; UNIX: AT&T Bell Laboratories; IBM PC: International Business Machines; MS-DOS: Microsoft; MC68000: Motorola; 8086/8087: Intel

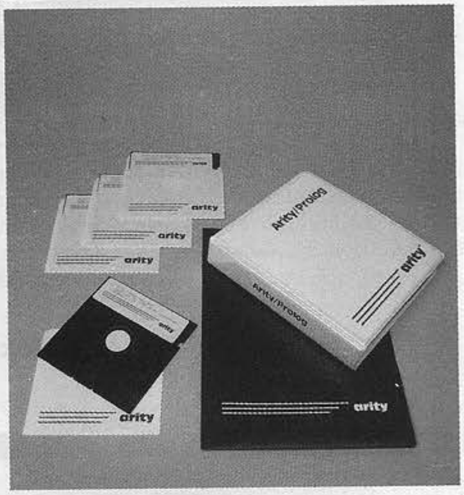
UniPress Software
Your Leading Source for UNIX Software.

CIRCLE 21 ON READER SERVICE CARD

More Power Than You Thought Possible

Arity offers the first serious implementation of Prolog for IBM personal computers. Arity/Prolog is a powerful, highly optimized, and extended version of the logic programming language Prolog. Imagine building software applications with a language that solves problems through deduction and logical inference. The task of creating complex programs is much faster and easier, resulting in lower development costs. Arity/Prolog is now in use in a wide range of applications in industry, business, research, and education. The solution—the *Arity/Prolog Interpreter:*

- Source level debugger
- Virtual databases, each with a workspace of 16 megabytes
- Floating-point arithmetic
- String support for efficient text handling



- Interface to assembly language and 'C'
- Text screen manipulation
- Integrated programming shell to MSDOS
- Comprehensive set of evaluable predicates
- Definite clause grammar support

Arity/Prolog Interpreter $495.00

Arity also offers the *Arity/Prolog Compiler and Interpreter,* a sophisticated development environment for building AI applications. Essential for producing fast, serious production code.

Arity/Prolog Compiler and Interpreter $1950.00

The *Arity/Prolog Demo Disk* is available for $19.95. ■ Arity/Prolog products run on the IBM PC, XT, AT, and all IBM compatibles. ■ To order, call (617) 371-2422 or use the order form below.

arity corporation 358 Baker Avenue, Concord, MA 01742

Name ____________________

Organization ____________________

Address ____________________

☐ Enclosed is a check or money order to Arity Corporation

☐ Please bill my ☐ Mastercard ☐ Visa ☐ American Express

Account # ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Valid ____/____ to ____/____ ____________________ signature

Quantity	Product	Unit Price	Total Price
	Arity/Prolog Compiler & Interpreter	$1950.00	
	Arity/Prolog Interpreter	$ 495.00	
	Arity/Prolog Demo Disk	$ 19.95	
		Subtotal	
		MA residents add 5% sales tax	
		Total Amount	

☐ Please send me more information about Arity and Arity/Prolog

arity 358 Baker Avenue, Concord, MA 01742

M-AD-08

CIRCLE 59 ON READER SERVICE CARD

Philippe Kahn: The man behind the Borland myth

By Regina Starr Ridley

[I] met a goddess, disk in hand,
There in Scotts Valley—home of Borland.

She sang to me with perfect pitch,
Of code that runs without a glitch.

Turbo Pascal; SideKick too,
And SuperKey, the one that's new.

Where Dionysian madness reigned,
Among those usually left-brained.

At what I heard my wonder grew,
For ancient Bacchus lives anew—

Reborn in this Age of Information,
As head of a Software Corporation.

—"Bacchus and The Myth of Borland"[1]

Philippe Kahn, founder and president of Borland International, can't find a bottle opener for the two bottles of Calistoga mineral water.

He improvises. He uses the doorjamb and flips the caps off. Mineral water sprays the air as if from a geyser, splattering his face, Hawaiian shirt, and tan corduroys. He continues talking without wiping his face, oblivious. He's not a man who worries about such things as appearances.

Kahn seems like a man who's living his dream. His two-year-old, privately owned company is bringing in an average of $2 million dollars a month with monthly sales of 60,000 units, which Kahn compares with Lotus Development Corp's 45,000 units.

In 1½ years, Borland has sold 400,000 copies of Turbo Pascal, a widely acclaimed piece of software originally priced at a revolutionary $49.95 (Turbo Pascal 3.0, released in March 1985, costs $69.95). Borland's second big product, the desktop organizer SideKick, has had sales of 300,000 units since June 1984. SuperKey, a data encryption and macro processor program, was released in April, and Borland says 15,000 orders were received in the first three weeks.

PC Week listed Turbo Pascal among the top 10 products of 1984, and *Infoworld* named SideKick product of the year in 1985. Borland was named company of the year in 1984 by *PC Magazine.*

Thirty-three-year-old Philippe Kahn holds the spotlight for this success. The media seems suddenly to have caught on to a hot personality. *The Wall Street Journal* covered Kahn in a front page story on June 6, playing up his maverick nature and flamboyant lifestyle under the headline, "L'Enfant Terrible is also L'Enfant Riche." Two days later—perhaps a typical day, perhaps not—Kahn participated in a morning radio talk show, at noon began a 2½-hour interview with *TIME* magazine on U.S. entrepreneurs, and immediately followed with the *COMPUTER LANGUAGE* interview.

"We'd like to get Philippe on the cover of *TIME* magazine," Borland's corporate communications director confides. Some might see an ego problem here, but that's an oversimplification. More accurately: Philippe Kahn is a very intelligent man who recognizes his abilities and has formed a company in his image. But there's more to the man than that.

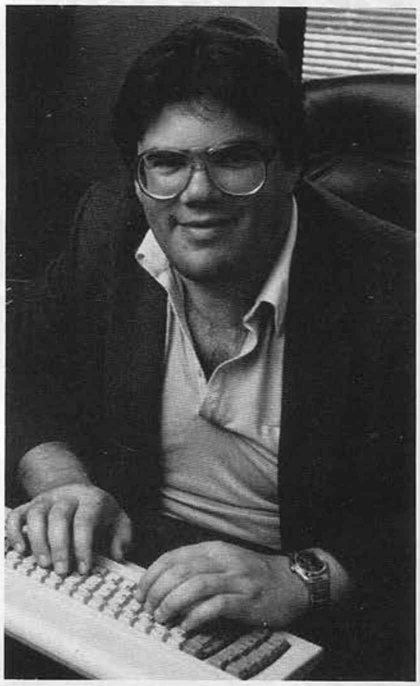
"You have to be careful of appearances, you know," Kahn says. "People and things are never as good as they appear to be. Sure, the company is super successful. I can be considered maybe even the most successful guy of the year, whatever, so what? There's more to it, there's more to someone than that."

Now Zeus, as you know, is fond of risk;
Of earthly maidens who boot the disk.

For if she smiles, Zeus can't refuse her,
And thus he consorted with the End-user.

Semele her name, the Market Place;
A princess full of riches and grace.

There in her womb grew the seed of Zeus,
Philippe, Bacchus, Ease of Use.



Kahn's university years were spent in France, where he was born, and Switzerland. He made money on the side playing his saxophone in French jazz clubs. His academic interest focused on theoretical mathematics and number theory, and he received an Agrégation de Mathématique, equivalent to a Ph.D. in mathematics, certified by the French Academy of Science. He taught university-level math in Nice and Grenoble, France.

"Computers were just a way for us to do applied mathematics and solve problems," says Kahn. "I was more interested in proving theorems and in the beauty of absolute pure mathematics done with just the mind and no machine."

"In the States computers at that time already had kind of an aura—there was another dimension than the pure scientific dimension, kind of like the dimension a rock group has to a high school class.

"The aura around the Apple is much more than just business and computers. It's something else. There is something a little mystical. But the mystique [in Europe] at that time didn't exist. The

---

[1] Copyright 1985 Jeffrey Armstrong.

GET ORGANIZED!

with

TURBO REF™

a Cross-Reference and Lister utility for Pascal source programs.

- All options user-selectable.
- Identifies line number for each variable reference.
- Can cross-reference constants.
- Indicates type of reference.
- Will process a list of files.
- Will read Include files.
- Draws boxes for control structures — example:

```
if PascalProgrammer then
  begin
    CallGracon;
    repeat
      RunTurboRef;
    until Organized;
  end;
```

Written in Turbo Pascal for the IBM PC.
PC-DOS 2.0 and 128K required.

Only $49.95

MC, Visa, or company check.
C.O.D. orders add $4.00.
Michigan residents add $2.00 sales tax.

GRACON SERVICES, INC.

4632 Okemos Rd.
Okemos, MI 48864
(517) 349-4900

IBM and PC-DOS, and Turbo Pascal are registered trademarks of International Business Machines Corp. and Borland International Inc., respectively.

CIRCLE 56 ON READER SERVICE CARD

punch cards . . . there wasn't much mystique there. For me, the mystique of manipulating ideas was far more attractive than the mystique of the punch card reader."

Kahn studied mathematics in Zurich, Switzerland, for a time. He happened to wind up in a class taught by Niklaus Wirth, champion of structured programming and author of Pascal and Modula. "At that time there was no such thing as a computer class, you had to learn a language," says Kahn. "As far as I can remember, there was a choice between PL/I and Pascal. When I went into the PL/I class there were 200 people. In the Pascal class there were five people, because no one knew anything about Pascal." Wirth was the professor, and Pascal was the first language Kahn learned.

What changed Kahn's mind about computers was his encounter with the video terminal and keyboard in the early 1970s. His first computer was the Apple II. "I discovered with my Apple II the mystique of Silicon Valley," says Kahn. "I thought the center of gravity of the high-tech world was Silicon Valley. It was an opinion, I'm not sure it was a fact. And it was a dream, you know, everyone has a dream."

In 1983, the Borland myth begins. Kahn founded Borland International that May, with a total of $20,000 in family investments. The first office was over an auto repair shop. Kahn chose the name Borland because it sounded very American to him—it was inspired by astronaut Frank Borman's name. Kahn didn't want to draw any attention to his immigrant status as an illegal alien (he now has a green card).

Turbo Pascal, Borland's first product, was released in November 1983. Its actual origins are shrouded in some mystery. Kahn says he and two friends—whose names he cannot (or will not) reveal—wrote it in Europe. They were making a living writing application code in Pascal for CP/M-80 machines.

But when the IBM PC arrived on the scene, "we found ourselves with the worst compilers for Pascal that have ever been written," Kahn says. "We really had a problem. We needed a Pascal compiler that behaved the way we wanted it to behave. First, we wanted it to be fast. Then we wanted it to be small. And we wanted it to generate good code. There came Turbo Pascal."

Then Bacchus, (Philippe) remembered and cried,
At how his Mother in sorrow had died.

How expensive programs dragged her away,
Making her suffer; making her pay.

"The User must have a product that's great!"
He shouted, descending to Pluto's Gate.

"Faster Compilers with vigor and speed,
Will give them the power and freedom they need."

So began their ecstatic quest,
To bring mankind programs, but only the best.

Products wrought in Fire Devine,
And priced at only Forty-nine.

Borland's philosophy of selling low-priced software is Philippe Kahn's philosophy. "Everyone was saying 'you should try to sell it [Turbo Pascal] for $400, $500.' One day I said 'hey, we're going to sell it for $49.95 plus $5 shipping and handling,' because I had bought a mail-order book at a B. Dalton's store, and the book said if you want to generate an impulse buy, the product has to be less than $50.

"That's how the whole concept came. And then the concept was such a success that it became a philosophy. When we saw the enthusiasm it generated, we said well, maybe we've got something here, something that everyone's been waiting for—super-quality software at rock-bottom prices."

Some other software publishers have been following Borland's lead in low software prices, notably Adam Osborne's Paperback Software. "I like Adam," says Kahn. "I hope that he will be successful because it will shut the mouths of all the people who have been putting him down for a long time."

"Regarding his company, I'm glad to see that people are following our lead in terms of pricing. Unfortunately, I'd hope that people would follow our lead in terms of quality, too."

Osborne's company markets software it purchases from a pool of software developers, who are then paid royalties. "I don't know how viable it is to do what he does and not develop your own software," says Kahn. "At our prices, you can't offer large royalties to people."

Non-copy protection is another one of Philippe Kahn's beliefs that became a Borland philosophy. Borland's policy of non-copy protection is held without regard to how it affects the company's sales.

"It is a philosophy," says Kahn. "I do not use copy-protected software. Why should I impose copy protection on my customers who are ready to spend money on a piece of software? Why should I tell them to do something I wouldn't do?"

"The problem with copy protection the way it is right now is that it's basically a tax on honesty. It's like saying, 'You're buying this thing and paying me money, but I don't trust you. You're thieves, so I'm copy protected.'

"I think it's a major problem to treat people that way. I think in any business relationship, if there's not at any point some sort of trust in the relationship, then I'm not interested. I'd rather go into the mountains and pick strawberries."

"For all mankind can be glad and hack
If we bring the excitement of programming back.

So let the taste of my programs flow;
Let ecstasy in their phosphor glow."

"The core of our corporate strategy is that we are a language company and that we know that languages are here for the long run. We aim to be the language company of the latter part of the eighties," proclaims Kahn.

Borland's choice of language products to develop over the next year or so ("I don't want to talk about time frames anymore" growls Kahn, "you know how software is—you start developing and then you pray") is Modula-2, C, BASIC, and then Ada.

"We chose Modula because I like Modula," says Kahn. He does not see a Borland Modula as a competitor to Turbo Pascal. "I believe that Pascal will probably remain the primary teaching language, as well as gradually supplant BASIC for reasonably small programs. I believe Modula-2 will be for people who want to go for bigger projects and have use for multitasking applications, because it does support quite nicely interprocess communications and things like that."

"It's like a toolbox, you know. If you start trying to use a saw as a hammer or a hammer as a saw you're in trouble. Why would people who want to make small projects, like printing a mailing list, use Modula-2 and have to hassle with some of the inconveniences of Modula-2 because it's a 'better' language?

"I don't think Modula-2 is a better language than Pascal, especially since Pascal has been extended. Now, if you're talking Modula-2 against standard Pascal, then O.K., sure. But a language like Turbo Pascal—probably the de facto standard Pascal—has a lot of the things that Modula-2 has in it, all the low-level stuff.

"Modula-2 will probably be bought by people who need the functionality of Modula-2," he says. "We know that Modula-2 will be far less successful than Turbo Pascal. We know it will be somewhat successful, but there is no way we can sell the same amount of Modula-2s and Pascals."

Borland's Modula-2 for CP/M-80 is currently in beta test. The PC-DOS version is in alpha test. The CP/M-80 Modula is further ahead because "one guy got excited about doing it for CP/M and was faster than the other guys," says Kahn.

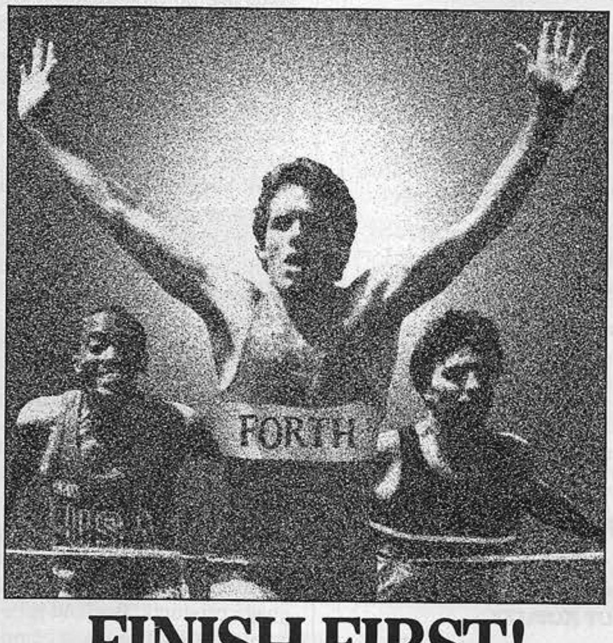
"It's a good compiler. It generates code that is two or three times faster than Turbo Pascal. It's the best CP/M-80 compiler.

"I think we're the only company developing new products for CP/M, and that's got to excite the whole CP/M community. We are a company that doesn't do things just because there's a market out there—and I think that's one of the keys to our success. We do things because we think they're going to be useful tools. If they're tools for us, they'll probably be tools for a lot of people, and then there'll be a market. That's how Turbo happened, that's how SideKick happened, that's how SuperKey happened, and that's how the CP/M version of Modula-2 happened.

"I'm sure we'll make money on the CP/M version of Modula-2. It's exciting for people to know there's a company that still cares about CP/M-80. And sure, we care because there are some guys who use CP/M-80; therefore we do products for it."

Kahn chose BASIC to be one of Borland's next compiler products even though he doesn't like BASIC. "The only thing I really needed was a BASIC compiler that could generate executable files out of a Microsoft 3.0 interpreter program," says Kahn. "And I thought that's what people needed. A machine that would allow you to run all those programs you find in the magazines or in the public domain. And eventually everyone will buy one even if they don't use BASIC, because everyone has a few BASIC programs that do something on their machine, whether it's a game or something like that."

START WITH FORTH...



FINISH FIRST!

polyFORTH II® Provides The Competitive Edge

polyFORTH II is the consistent winner in any race. Especially one that needs the optimum software environment—language and operating system—for interactive and real-time applications. Now, you can program quickly and efficiently for process control, graphics, instrumentation, and robotics using the English words common to the application.

The gold-medal award winning features of polyFORTH II include:

☑ unlimited control tasks;

☑ high-speed interrupt handling;

☑ reduced application-development times; and

☑ ability to support multiple terminals simultaneously

and they're available for mini or microcomputers like the IBM-PC and the DEC PDP-11/LSI-11 and 68000.

FORTH, Inc. supports many aerospace giants and Fortune 500 companies. These are the companies that finish first consistently, because they execute their jobs with Forth as their special "secret weapon" for real-time applications and manufacturing control.

For over a decade, FORTH, Inc. as the inventor of the Forth environment, has provided users with extensive on-line documentation; complete, user-friendly manuals; programming courses; a FORTH, Inc. hot-line; and expert contract programming and consulting services. And all this at a surprisingly low cost.

Make polyFORTH II part of your medal-winning strategy as you go for the gold in your next race....

FORTH, Inc.
2309 Pacific Coast Highway
Hermosa Beach, California 90254
(213) 372-8493 • TELEX (RCA)275182

IBM-PC is a registered Trademark of International Business Machines Corporation.

CIRCLE 30 ON READER SERVICE CARD

EXSYS

A LOW COST EXPERT SYSTEM DEVELOPMENT TOOL

Create knowledge based, probabilistic IF-THEN rule expert systems on an IBM PC, XT or compatible

Provides the user with a list of possible solutions to a problem arranged in order of probability

Performs backward chaining and can explain how conclusions were reached and why information is needed

External programs, including most spread sheets and data base managers, can be called with data passed into EXSYS for analysis

ALL input is English text, algebraic expression or menu selection

Numeric variables are supported and can be calculated and displayed by the expert system

Full available memory is used and the programs are written in C for fast, efficient code

700 rules per 64k of memory over 192k (almost 5000 rules in a 640k PC)

Trig, log, exp and sqrt functions supported

Developed expert systems marketable **WITHOUT ROYALTY**

Requires IBM PC, XT, AT or compatible with DOS 2.0 or higher and 256k

DEMO DISK — Demo knowledge bases, text of manual, automated instruction on how to use the program and ability to create and run rules – $10 (refundable with purchase)

FULL PROGRAM — $295 + $5 postage and handling

MACINTOSH VERSION AVAILABLE SOON

EXSYS INC, P.O. BOX 75158
CONTR. STA. 14, ALBUQUERQUE, NM 87194
(505) 836-6676

CIRCLE 54 ON READER SERVICE CARD

In the past Kahn has been quite vocal about his distain for C. He's only mellowed a bit, but C is high up on the Borland product list.

"If people want a C compiler, we'll give them a C compiler. We're a language company, and we have the technology to produce very good compilers. Producing a C compiler is essentially writing a slightly different parser than Modula-2's, because the functionality of C is very close to that of Modula-2."

Kahn says he uses C for certain things, such as drivers. He's working on a Sun terminal and uses C on it because he doesn't think its Pascal compiler is very good.

"But I personally don't like to take C and write a program," says Kahn. "I like to take Pascal and Modula-2 and fiddle around, write little games, have fun on a plane with my DG while people are watching 'Karate Kid' or something. Write a little Space Invaders game, something useful, or something funny."

"Do you know what C is? There are two types of languages. C is a write-only language. Pascal is a read and write language. Take two 1,000-line programs. One in Pascal, one in C. You didn't write them, someone gave them to you. You know both languages well. With the Pascal program, after 10 minutes maximum you'll know what it does. With the C program, you could spend two weeks figuring out what it does. It's a write-only language.

"I think the dazzling penetration of C among a certain fringe of programmers is due to the craze over UNIX. Everybody said 'UNIX is going to be successful, therefore C is the language.' Well, everybody knows by now that UNIX won't be successful, and therefore C probably will be an interesting language in certain areas, but won't be 'the' language."

After Modula-2, BASIC, and C language products, Borland is considering coming out with an Ada compiler. Kahn plans to implement a reasonable subset of Ada for people who want a fast, efficient development system. "The goal here is that if people develop software with our compiler, then once they finish, when it works, then they can compile it on a validated Ada compiler. And then they have a really easy way to work," says Kahn.

Kahn has no plans for COBOL, Forth, or AI language products. "We do what we like to do, and I never really liked COBOL. Unless someone comes on with the company who just loves COBOL, wants to do a Turbo COBOL or something, I just won't get involved with it."

"Forth is a religion, and we've kind of decided right now not to participate in religion, one side or the other.

"I think AI is a big bunch of baloney. What they call AI is simply manipulating more data in less time. AI is not much more than glorified Animal programs.

"AI languages are interesting. LISP is an interesting language. PROLOG is probably even a more interesting language. A language like Smalltalk is cute. I like it, I think it's nice."

Kahn sees Borland as a fun place to work. And judging from the parties, it is. The story of the Borland toga party at the 1985 West Coast Computer Faire in San Francisco, Calif., received a great deal of press coverage. About 600 Borland employees and guests—most wearing grape leaves in their hair—participated in an unparalleled spectacle of a party.

Philippe—Bacchus—wore a toga and played his sax. A troupe of San Franciscan clowns, called I Fratelli Bologna, led high-tech versions of Olympic games, such as throwing the diskette. Other entertainment included a stunning male trapeze artist and a rock band. Meanwhile, party-goers feasted on an outrageous buffet— which included a whole roasted lamb and pig—and enjoyed the flowing spirits. Borland's cost: $45,000.

Kahn says Borland employees have fun at work too. "If you're a software developer at Borland you're proud of what you're doing because we don't have 'junk products.' There's no product we're ashamed of. And that's important. That's a craftsman's pride."

Now each day they're growing and all through
the land,
Users are singing the praises of Borland.

Yes born of the Gods but destined for men,
To put the job back in computing again,

That programs divine for any PC,
Delight every user, priced reasonably.

So ring forth the Music, the Timbrel and Flute,
Take off your Wing-tips and Three-piece suit,

And join in the merriment, sit back, relax,
Here's Borland as Pan and Philippe on
the Sax.

Regina Starr Ridley is the Managing Editor of COMPUTER LANGUAGE.

Microprocessor programming made simple.

"Keep it simple" was the principle of the 14th Century English philosopher William of Occam and it has even more validity today. Faced with the problems of sophisticated computer systems, designers have found that ever more complex programming languages are further complicating their tasks. Until now.

Occam. Created for system design and implementation.
When we started designing our new VLSI family of 10-MIP transputers, we built on William's simple philosophy. To take advantage of the possibilities opened up by the transputer, we needed to create a language capable of properly addressing parallelism and multiprocessor systems.

With the ability to describe concurrency (whether timeshared or real) and to handle message-passing at the lowest level of the language, all aspects of a system can be described, designed and implemented in occam. From interrupt handling through signal processing to screen editors to artificial intelligence. And on.

But occam is not limited to our transputer family. It provides an efficient, responsive implementation language for systems built on today's microprocessors. It also opens up future possibilities with its performance-enhancing multiprocessor capabilities. And INMOS now offers a product to let you exploit occam's total capability in your system.

Simplify your job with the Occam Programming System.
The Occam Programming System (OPS) gives you the tools for complete VAX / VMS software development. This package includes an integrated editor / checker, an optimizing VAX compiler and full documentation. This gives you a supportive environment for the development of occam programs for execution on the VAX. Cross-compilers for 68000 and 8086-based systems will also be available.

What's more, the occam programs developed and proven on the OPS will give you a head start for work with the INMOS transputer. Extensions to the OPS will be available which will allow occam programs to run on the transputer.

And if you have a requirement to program the transputer in other popular high-level languages, other extensions will include compilers for C, Fortran, and Pascal.

Get started today.
Contact us for our information pack on occam, the Occam Programming System and the transputer. You'll be surprised how simple your life can be.

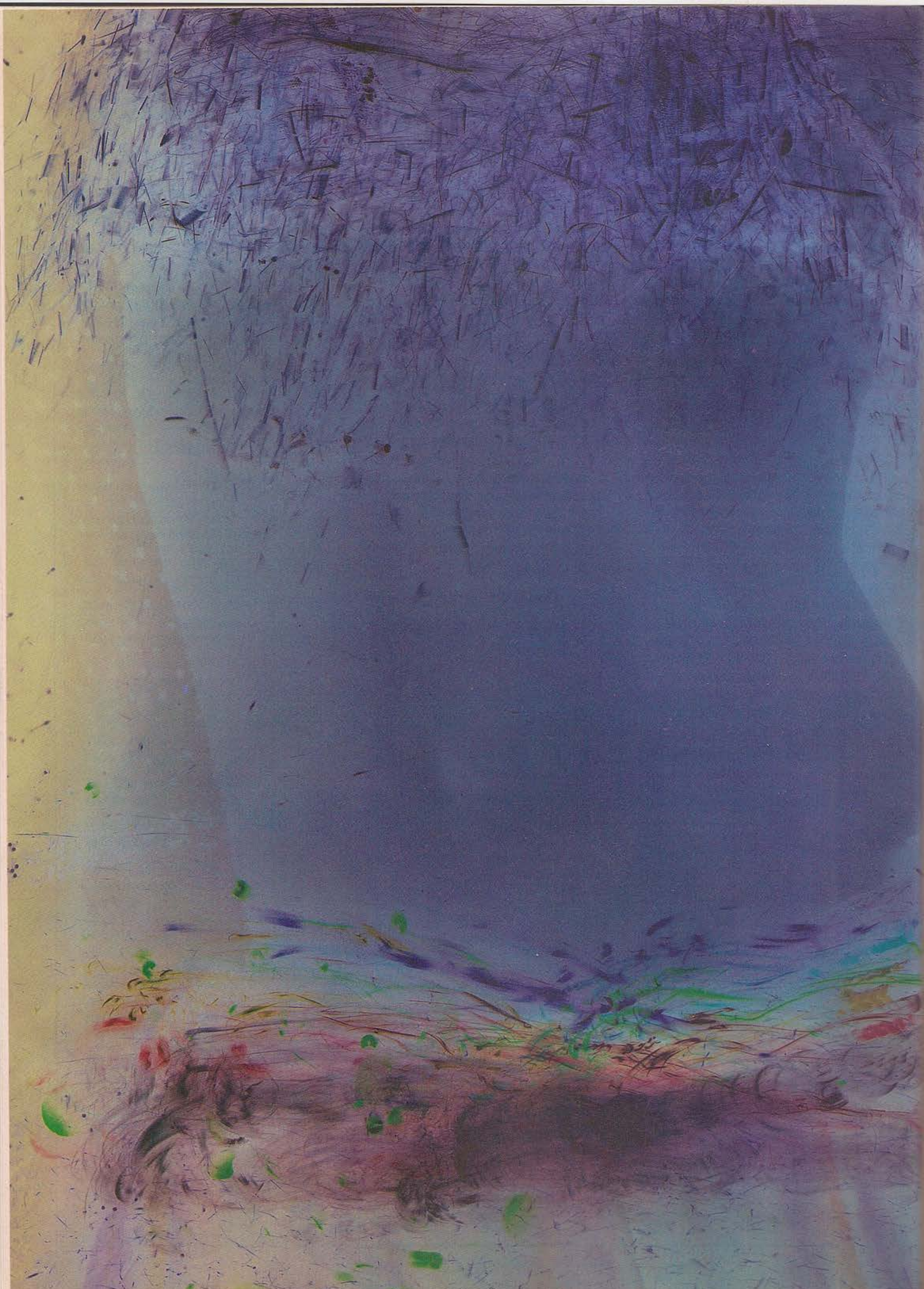
For quick response, call us at (303) 630-4000 or write:
Occam, P.O. Box 16000,
Colorado Springs, CO 80935.

inmos

inmos, and occam are trademarks of the Inmos Group of Companies

FORTRAN
COBOL
BASIC
APL
PL/I
C
PASCAL
ADA
occam

CIRCLE 32 ON READER SERVICE CARD



Smalltalk Comes to the Micro

By Darryl Rubin

Ten years before Apple's Macintosh opened its first window, researchers at Xerox Corp. were using personal computers that sported a what-you-see-is-what-you-get, point-and-click user interface with bit-mapped graphics, overlappiing windows, and pop-up menus.

These minicomputer-sized "PCs" were called Altos, and they were programmed in a language called Smalltalk.

Smalltalk pioneered many of today's most important software concepts, including object-oriented programming, message passing semantics, and modeless design. The most recent version of the system, Smalltalk-80, was also an unprecedentedly accessible one; it was written almost entirely in itself and provided multiwindowed "browsers" through which you could inspect or extend the system. And extending the system was easy, because by its very design principles, the Smalltalk-80 system encouraged exploratory programming and a toolkit approach to software design and reuse.

Not long ago, if you wanted to explore a Smalltalk-80 system you had to work at Xerox—no commercial implementations existed. But that situation is changing, and at least one product is now available for personal computers: Methods, from Digitalk Inc. Methods implements a large subset of the Smalltalk-80 system as described in *Smalltalk-80: The Language and its Implementation,* by Adele Goldberg and David Robson (Addison Wesley, 1983).

In this article we'll explore the Smalltalk-80 language and learn how to write Smalltalk-80 programs. As programming examples, we'll present some surprisingly short routines that implement UNIX-style filters and even an interactive spelling checker. We'll also highlight the differences between "pure" Smalltalk-80 and the Methods implementation of it.

For a quick intro to Smalltalk-80 syntax, check out the window labeled Language Comparison in Figure 1. This and the other figures and listings we'll present were all developed on an IBM PC/AT running Methods. If you have your own copy of the program, fire it up and come explore!

Object lesson

A Smalltalk-80 system is made of objects that send messages to each other. Objects include everything from numbers and characters to arrays, sets, files, windows, code blocks, and text editors. Each distinct kind of object is called a class. An object that belongs to a particular class is called an instance of that class. Some examples of objects and their classes are:

29	Integer
$A	Character (A)
'ABC'	String
#Smalltalk	Symbol (unique string)
#(1 2 3)	Array (of integers)
#(1 $A 'BC')	Array (of various)
[x := 1 + 2]	Block of code

More complex objects are the text editor labeled *Workspace* in Figure 1 and the *Class Hierarchy Browser* in Figure 2. Of course, you can add your own classes of objects to a Smalltalk-80 system. To write a Star Trek game, for example, you might first create a *Starship* class, then create an instance of that class called *Enterprise*.

All computations in Smalltalk-80 are performed by objects. Indeed, objects are like little computers: each stores some data plus the procedures that operate on that data. To access or manipulate the data stored in an object, you send it a message detailing your request. In response, the object executes the internal program selected by your message and returns the result (another object, naturally).

An object's stored data, called its instance variables, serves to describe the current value or state of the object. For example, *Integer* objects have a single instance variable that specifies the numeric value of the integer. *Starship* objects would probably be more complex, having instance variables for the *Starship*'s current speed, heading, fuel, and damage level.

An object's internal programs, called its methods, serve to implement the behavior or functionality of the object. The code for these methods is contained in the class definition for the object. For example, objects of class *Integer* have methods for performing arithmetic, numerical comparison, and type conversion, among others. *Starship* objects might have methods for changing speed, turning, raising shields, refueling, and so on.

If you think Smalltalk-80's encapsulation of data and procedures inside

objects smacks of Ada's packages or Modula's modules, you're right. It's also what makes all three languages so modular and extensible, because it prevents software components from depending on each others' internal representations or implementations. Smalltalk-80 smacks of Ada in another interesting way, thanks to its message passing semantics, as we'll see later.

The message is the medium

All computation in Smalltalk-80 occurs as a result of sending messages to objects. For example, the expression *2 + 3* causes the message " + " (called the message selector) to be sent to the object *2* (the message receiver), with the object *3* as a message parameter. The object *2* responds to the " + " message as you would expect: by adding its own value to that of the parameter and returning the result, *5*, as another object.

The *Workspace* window in Figure 1 shows several kinds of message expressions and the results they return. The Smalltalk-80 system is unlike others in the flexibility it gives you for entering and evaluating expressions. You can type expressions anywhere in almost any window on the screen, scrolling the window contents in any direction if you need to.

To evaluate the expression, you highlight it with the cursor and pick the command *Show It* from a pop-up menu. Just like that, the answer prints out adjacent to the highlighted expression. (Or, if the expression has an error, an error message is inserted into the expression adjacent to the error.)

There are three kinds of messages: unary, binary, and keyword. A unary message consists of a message name with no arguments, as in:

```
'ABCD' size    4
```

Here, the unary message *size* is sent to a string object, which responds by returning the string's length. The *size* message is quite a common one in Smalltalk-80; almost every kind of object implements its. Array objects, for example, respond to *size* with the number of cells they have.

Binary messages take a single argument and are written in infix notation, as follows:

```
2 + 3           5
'ab' > 'cde'   false
'ab'~= 'cde'   true
```

Keyword messages take one or more arguments and use message selectors that have trailing colons. An example of a one-argument keyword message is *2 max: 3*, which returns the object *3*. Here is a two-argument message:

```
Prompter prompt: 'Suspect word'
     default: 'hee*'
```

This message sends the selector *prompt:default:* to the object *Prompter*, with the two strings as arguments. The result is the pop-up prompter shown in Figure 2. You build messages with three or more arguments in a similar manner.

Smalltalk-80 lets you build arbitrarily complex message expressions from simpler ones. The language has few precedence rules. Terms of equal precedence are evaluated left to right, with parenthesized expressions being evaluated first. Unary messages are sent first, binary messages next, and keyword messages last. So the expression:

```
anArray at: aSymbol hash
      put: aSymbol size + 1 * 2
```

Smalltalk-80 syntax and sample expressions

Language Comparison

```
PASCAL                   SMALLTALK-80

x : = 2 * (y + z);       x : = 2 * (y + z).
a[i] : = a[j];           a at: i put: (a at: j).
x : = max (y,100);       x : = y max: 100.

s : = 'string';          s : = 'string'.
l : = length(s);         l : = s size .
t : = concat(s,s);       t : = s , s.
u : = copy(t,1,l);       u : = t copyFrom: 1 to: l.

new (aSet);              aSet : = Set new.
aSet : = aSet + [2];     aSet add: 2.

if [n] in aSet           (aSet includes: n)
  then n : = log(n)        ifTrue:  [n : = n log]
  else n : = abs(n);       ifFalse: [n : = n abs].

for i : = 1 to 10        1 to: 10 do:
  do a[i] : = 0;           [:i | a at: i put: 0].

while not(eof(f))        [f atEnd] whileFalse:
  do read(f,ch);           [ch : = f next].
```

Workspace

EXPRESSION	RESULT
− 2 * − 3 abs	− 6
('ab' , 'de') size	4
$* isLetter	false
'abc' asUpperCase	'ABC'
'abc' at: 2	$b
#(1 2 $a 'b') at: 4	'b'
"Assume an empty set Myset"	
Myset add: 1	1
Myset add: $a	$a
Myset includes: 2	false
Myset	Set($a 1)
"Assume an empty dictionary D"	
D at: 'yes' put: 'si'	'si'
D at: 'yes'	'si'
D keyAtValue: 'si'	'yes'
D includes: 'si'	true
D includes: 'yes'	false
D includesKey: 'yes'	true
D at: 8 put: 10	10

Figure 1.

is evaluated as if it had been written:

```
anArray at: (aSymbol hash)
     put: (((aSymbol size) + 1 ) * 2)
```

Here's an example that shows when parentheses are a must:

```
aSet add: (anArray at: 1)
```

Had this expression been written without parentheses, the invalid message selector *add:at:* would have been sent to the object *aSet* with *anArray* and *1* as arguments, hardly what we intended!

If you need to send several messages to the same object, you can cascade them with semicolons, like this:

```
Enterprise turn: 90;
     ahead: 5; shieldsUp
```

You can also assign the results of an expression to a named variable, as in *aDict := Dictionary new* (here, as in Methods, we use the symbol ":=" rather than Smalltalk-80's left arrow). This creates a new dictionary and assigns it to the symbol *aDict*, which will now serve as a name for that object in any expression. So now you can write:

```
aDict at: 'yes' put: 'si'   "Store"
aDict at: 'yes'             "Look up"
```

Perhaps you noticed that dictionaries and arrays use the same messages, or protocol, for storing (*at:put:*) and retrieving (*at:*) elements. Indeed, objects have a great deal of commonality in the messages they implement. This provides a consistent and type-independent way to deal with many kinds of objects, be they numbers, strings, arrays, sets, or display windows. Type independence is further promoted by the fact that Smalltalk-80 implements dynamic binding, meaning that a message specifies the name rather than address of the desired operation.

These characteristics make it possible to write general-purpose routines akin to Ada's generic procedures. Suppose you write a Bubble Sort method, using the message "<" for comparisons. Then your method will be able to sort any kind of object that implements "<", including numbers, strings, and dates. Write a LIFO stacking method, and this one routine will be able to stack numbers, arrays, sets, windows, even starships!

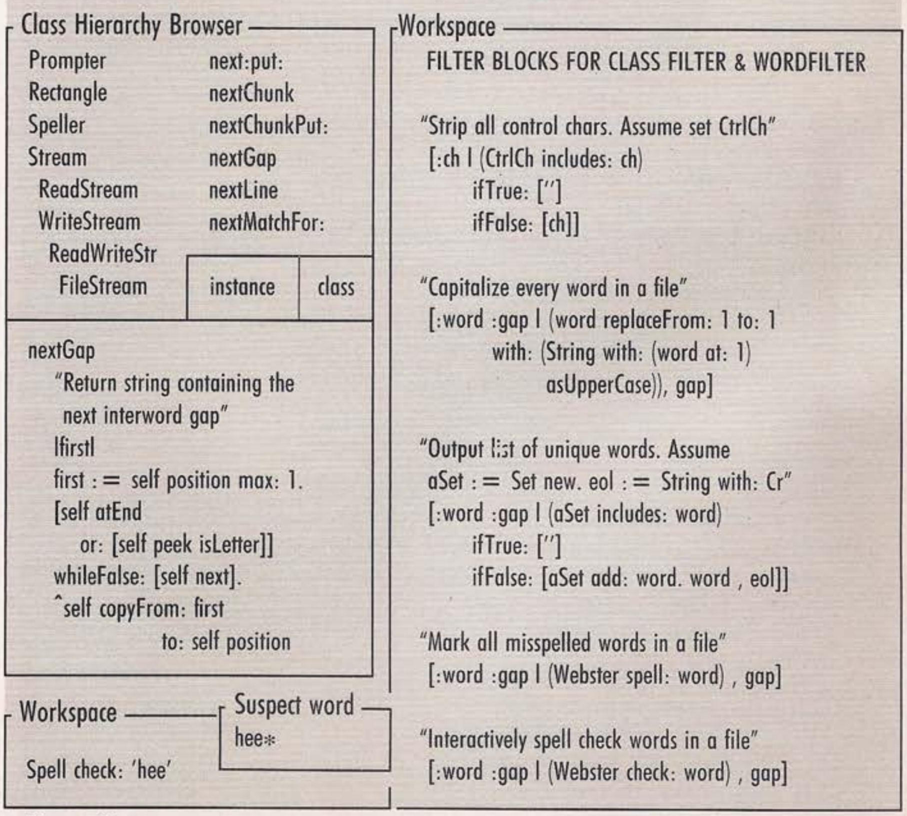
Class act

A Smalltalk-80 system consists of many kinds, or classes, of objects, each of which performs different functions. *Integers, Strings, Sets,* and *Dictionaries* are some of the standard classes. All objects of the same class have the same structure: the same instance variables, message protocol, and methods. For example, all *Integer* objects have an instance variable for the integer's value, and all respond to messages for performing arithmetic and type conversion.

Smalltalk-80's class structure is hierarchical, consisting of a root class, called *Object*, and many subclasses. Class *Object* provides the common protocol for all objects, including standard messages for testing an object's class (*isKindOf: aClass*), copying the object (*copy*), or printing it out symbolically (*printOn: aStream*). Each subclass adds to this its own messages and methods for implementing the subclass-specific functionality.

The most significant feature of Smalltalk-80's class system is that each class inherits the entire functionality of all

nextGap, sample Filter blocks, and a Spell window



```
nextGap
   "Return string containing the
    next interword gap"
   |first|
   first := self position max: 1.
   [self atEnd
      or: [self peek isLetter]]
   whileFalse: [self next].
   ^self copyFrom: first
              to: self position
```

```
Spell check: 'hee'
```

```
hee*
```

```
FILTER BLOCKS FOR CLASS FILTER & WORDFILTER

"Strip all control chars. Assume set CtrlCh"
 [:ch | (CtrlCh includes: ch)
      ifTrue: ['']
      ifFalse: [ch]]

"Capitalize every word in a file"
 [:word :gap | (word replaceFrom: 1 to: 1
         with: (String with: (word at: 1)
               asUpperCase)), gap]

"Output list of unique words. Assume
 aSet := Set new. eol := String with: Cr"
 [:word :gap | (aSet includes: word)
      ifTrue: ['']
      ifFalse: [aSet add: word. word , eol]]

"Mark all misspelled words in a file"
 [:word :gap | (Webster spell: word) , gap]

"Interactively spell check words in a file"
 [:word :gap | (Webster check: word) , gap]
```

Figure 2.

its superclasses, straight up to class *Object*. This means you can add new classes to the system simply by changing or extending an existing class to create a subclass of it. You never need reinvent an existing piece of functionality. You are also free to modify existing classes, because Smalltalk-80 is written almost entirely in itself.

Like everything else in a Smalltalk-80 system, adding or modifying classes is done through windows. The *Class Hierarchy Browser* in Figure 2 shows one such window for doing this. Clockwise from the top left corner, this window shows a class pane, message pane, and method pane. To modify an existing class, all you do is select a class name from the class pane, then select one of the class's messages from the message pane, and finally edit the method that appears in the method pane.

```
Filter class definition and methods

Object subclass: #Filter
  instanceVariableNames:
    'filterBlock '        "Code that does the filtering"
  classVariableNames: ''  "None"
  poolDictionaries: ''    "None"

Filter class methods

using: aFilterBlock

  "Create a new filter using the specified block."
   ^self new using: aFilterBlock

Filter methods

from: inStream to: outStream

  "Filter from input to output stream, returning output
   stream."
   inStream do: [:item |   "Do for each item in inStream"
     ((item := filterBlock value: item) isKindOf: String)
        ifTrue:  [outStream nextPutAll: item]
        ifFalse: [outStream nextPut: item]].
   ^outStream.

fromFile: inFile toFile: outFile

  "Filter from the specified input file and append to the
   output file. Return the output file name."
  ^(self        "Invoke self with files opened as streams"
    from: (Disk file: inFile)
    to:   (Disk file: outFile) setToEnd)
   close file name.

using: aFilterBlock

  "Make the specified block be this object's filter block."
   filterBlock := aFilterBlock
```

Listing 1.

You can also add methods or classes by selecting those respective functions from pop-up menus associated with browser windows. Figure 2 shows an example of my having done exactly that: the *nextGap* method is one that I added to class *Stream*. It complements the standard *nextWord* method and is useful in writing UNIX-like filters that process streams a word at a time, as we'll see.

When you add a class to the system, you must specify several things:

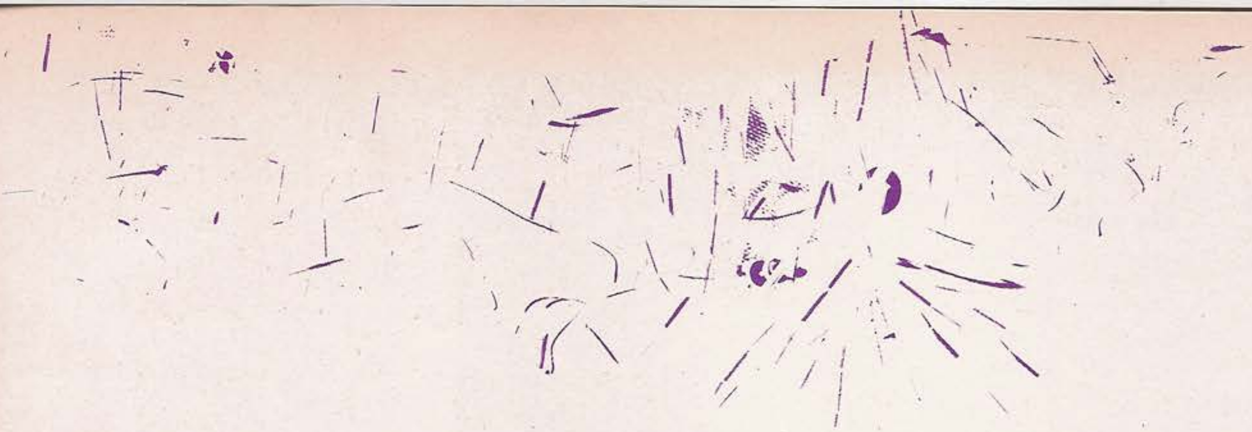
- The names of the new class and its parent class.
- The names of the instance variables for the new class. Each object of the class will get a private copy of these variables.
- The names of the class variables for the new class. The class variables are shared by all objects of the class.
- The message selectors for the new class.
- The methods for each defined message selector.

Once you've entered this information, Smalltalk-80 compiles it into the system image, and it's ready for use. Listing 1 shows a class that I added to my copy of Methods. The class, called *Filter*, implements a UNIX-like filter capability. Here's a simple example of using a filter to convert a file to uppercase:

```
(Filter using:
    [:ch | ch asUpperCase])
fromFile:  'in.txt'
toFile:    'out.txt'
```

This expression first creates a new *Filter* object by sending the block of code *[:ch | ch asUpperCase]* to the *Filter* class (we'll describe blocks in the next section). The *Filter* class stores that block as an instance variable of the newly created filter object. The new filter is next sent the *fromFile:toFile:* message, which specifies input and output file names for the filter to use. The result is to read the input file a character at a time, sending each character to the stored block and writing the block's resulting value to the output file.

Sometimes it's useful to process a file a word rather than character at a time. For this purpose I created a subclass of *Filter* called *WordFilter.* The definition of *WordFilter* (Listing 2) consists of but a single method that implements the message *from:to*. This message overrides a mes-



sage of the same name defined in class *Filter*. The difference is that *WordFilter* reads its input stream a whole word at a time and sends two arguments to the stored filter block: the input word and the interword gap (white space) that follows the word in the input stream. To accomplish this, notice how *WordFilter* uses the *nextGap* method (Figure 2) that I added to class *Stream*. Here's how to convert a file to upper case using a word filter:

```
(WordFilter using:
    [:word :gap |
      word asUpperCase , gap])
fromFile:  'in.txt'
toFile:    'out.txt'
```

WordFilter demonstrates the true power of Smalltalk-80's class inheritance mechanism because *WordFilter* inherits and uses all of *Filter*'s messages and methods except *from:to:*, which it overrides. For example, when you evaluate the preceding expression, Smalltalk-80 searches the *WordFilter* class definition for the *using:* message. *WordFilter* doesn't implement this message, so Smalltalk-80 searches upward through the class hierarchy for the first class that does implement the message, which happens to be *Filter*. *Filter*'s method for *using:* is therefore executed. Ditto for all the other messages that *WordFilter* doesn't itself implement.

You can do quite a lot with *Filter*s and *WordFilter*s. The *Workspace* window in Figure 2 presents some blocks that you can plug into the previous filter expressions to perform various operations. The brevity of the code to do such jobs could make even a UNIX wizard envious!

Method actor

Methods are what perform all the actions in a Smalltalk-80 system. In fact, almost all the Smalltalk-80 system code is implemented as methods and any software you add to the system will be in the form of methods, too.

Analogous to function calls in other languages, methods can take zero or more arguments and always return a value. They can also declare local variables that are created anew each time the method executes.

You can add methods to any class in the system—none are sacred. Simply display the desired class in a browser and pick *Add method* from a pop-up menu. This puts you into the browser's method pane where you use Smalltalk-80's excellent text editing functions to enter the method. The *nextGap* method in Figure 2 is a perfect example. It shows that a method consists of four things:

- The message pattern for the method. This is the name of the message, if unary (for example, *nextGap*), or the name of each keyword and its argument (for example, *fromFile: aFile toFile: aFile*).
- An optional declaration of temporary variables (for example, | *first* |, | *x y* |).
- A sequence of one or more expressions to be executed, with each expression delimited by a period.
- An optional return expression prefaced by an up arrow (^) that specifies the object to be returned as the value of the method. If omitted, the object executing the method (*self*) is returned.

The *nextGap* method also illustrates another important Smalltalk-80 programming concept: the importance of *self*. *Self* is a pseudovariable that lets an object send messages to itself, as when one of its methods needs to call another. In Figure 2, *nextGap* uses *self* to call two of the other methods defined in class *Stream*: *position* and *copyFrom:to:*. The more you get into Smalltalk-80, the more you'll notice objects talking to themselves!

To really get into methods, you need to understand Smalltalk-80's control structures. Like everything else, these are implemented as messages sent to objects. A simple example is the repeat loop:

```
10 timesRepeat: [Enterprise fire]
```

Integer objects respond to the *timesRepeat:* message by evaluating the bracketed expression a number of times equal to their value. Another pair of messages they respond to are *to:do:* and *to:by:do:*, which implement what you Pascal programmers would call a *For* loop (Figure 1 has an example).

```
WordFilter class definition and methods

Filter subclass: #WordFilter
  instanceVariableNames: ''   "inherited from Filter"
  classVariableNames: ''
  poolDictionaries: ''

WordFilter class methods

  "Inherited from Filter"


WordFilter methods

from: inStream to: outStream

  "Filter words from input to output stream, returning
   output stream."
   [inStream atEnd] whileFalse:
     [outStream nextPutAll:
        (filterBlock value: inStream nextWord
                     value: inStream nextGap)].
  ^outStream.
```

Listing 2.

```
Speller Class Definition and Methods

Object subclass: #Speller
  instanceVariableNames:
    'wordList correctList ' "Valid word list and autocorrect list"
  classVariableNames:
    'Marker'                "Char for marking misspelled words"
  poolDictionaries: ''

Speller class methods

words: wordFile corrections: correctFile

  "Create a new speller object, reading the word list
   from wordFile and correction list from correctFile.
   Also initialize the character that misspelled words
   are marked with."
   Marker := $*.
   ^self new initialize
      words: wordFile;
      corrections: correctFile


Speller methods

addCorrections: inStream

  "Add correction word-pairs from the input stream to
   the correction list.  Each word pair is on its own line.
   A line with only one word specifies a word to be deleted
   rather than replaced."
   |entry word|
   [inStream atEnd]
     whileFalse:  [entry := ReadStream on: inStream nextLine.
                   word := entry nextWord.
                   entry nextGap.
                   entry atEnd
                     ifTrue:  [self correct: word with: '']
                     ifFalse: [self correct: word
                                        with: entry nextWord]]

addWord: aWord

  "Add a word to the word list."
   aWord notNil
     ifTrue: [wordList add: aWord trimBlanks asUpperCase]

addWords: inStream

  "Add words in the input stream to the word list"
  [inStream atEnd]
    whileFalse: [self addWord: inStream nextWord].
  self addWord: ''

check: aWord
```

Listing 3. *(Continued on following page)*

Strange as it may seem, the bracketed expressions used in control structures are actually objects—objects of class *Block*. Whereas Smalltalk-80 responds to unbracketed expressions by evaluating them, it responds to bracketed ones by instantiating them as block objects.

Blocks serve as containers for message expressions that are not to be executed until a later time. When the time is right, sending the block the message *value* causes its contents to be evaluated and a result returned. For example, in the previous expression, the integer 10 sends the argument block the message *value* 10 times. You can also pass arguments to blocks by using the messages *value:*, *value:value:*, etc., as in:

```
[:str | str size] value: 'myString'.
avg := [:n1 :n2 | (n1 + n2) / 2].
avg value: 2 value: 4
```

Blocks give great power to Smalltalk-80 because they allow methods to pass code as arguments to other methods or even to synthesize code on the fly. This makes Smalltalk-80 suitable for AI programming.

Although not exactly an example of AI, Listing 3 does present a surprisingly small yet functional spelling checker, complete with pop-up prompters and an automatic misspelling correction feature! The comments in the first five methods give you the details. Here's how you would create a *Speller*, with the dictionary and correction list loaded from disk:

```
Webster := Speller
    words: 'words.lst'
    corrections: 'corrs.lst'.
```

To spell check a single word, you could now evaluate the expression *Webster check: 'hee'*. The result is the prompter shown in Figure 2. If you'd like to spell check an entire file, look just to the right of the prompter. What you see is the filter block you'd pass to *WordFilter* in a *using:* message (see last section). Yes, *Speller*s are actually designed to be used with *WordFilter*s; because of this, they inherit all the generality (and code!) that filters have to offer.

The Smalltalk of Digitalk

Digitalk's version of Smalltalk-80 may surprise you. It does what many, including myself, weren't sure was possible. Here is a personal computer implementation of not just the Smalltalk-80 language, but also much of the Smalltalk-80 system—pop-up windows, browsers, and all. Here is an implementation that is faithful to the spirit of a system implemented in its own language, with all the source code accessible to every end user (almost 200K worth). Here is an implementation whose performance is adequate for serious use.

Of course, personal computers are not (yet) Xerox Altos, so Methods does betray some disappointing, though reasonable compromises. There are no bit-mapped graphics or mouse support, two of Smalltalk-80's hallmarks. Instead, you get character mapped equivalents to Smalltalk-80's *Bitblt, Point,* and *Rectangle* classes, but with color thrown in (something the Alto didn't have). As for the mouse, Digitalk's way of using the numeric keypad makes for a serviceable alternative, but still . . .

You'll also find that Methods' classes and message protocols are not always identical to Smalltalk-80's. *Views* are called *Panes*, *Controllers* are called *Dispatchers*, the file system is different (thank goodness—it uses standard DOS files and directories). You get walkbacks, but no debugger. And no multitasking.

I spent a long time looking for other differences and discovered something else. You can spend months browsing the Methods source code and still discover a useful new method at every turn, or think of one you'd like to add, and can, on the spot.

Methods, like a Smalltalk-80 system should be, is yours to explore, to mold, and to master. ■

Methods $250
Digitalk Inc.
5200 West Century Blvd.
Los Angeles, Calif. 90045
(213) 645-1082

Darryl Rubin is section manager for network products at ROLM Corp.

```
  "Spell check a word and if misspelled, prompt the
   user with the marked word.  The user may hit CR
   to output the marked word, erase the mark to accept
   the word as is, or edit the word to correct it.  If
   the user accepts or edits the word, the wordList and
   correctList are updated accordingly."
   |word|
   word := self spell: aWord.
   (self isMarked: word)
   ifTrue: [word := Prompter prompt: 'Suspect word'
                             default: word.
            (self isMarked: word)
            ifFalse: [self addWord: word.
                     word ~= aWord
                     ifTrue: [self correct: aWord
                                      with: word]]].
   ^word

correct: badWord with: goodWord

  "Add a word pair to the correction list."
   ((badWord notNil) and: [goodWord notNil])
     ifTrue: [correctList at:  badWord trimBlanks asUpperCase
                         put: goodWord trimBlanks asUpperCase]

corrections: correctFile

  "Add correction pairs from the specified file to the
   correction list."
   self addCorrections: (Disk file: correctFile)

initialize

  "Create word list and correction list objects for speller"
   wordList := Set new.
   correctList := Dictionary new.

isMarked: aWord

  "Return true if aWord has been marked by the spell method"
   ^(aWord size > 0) and: [(aWord at: aWord size) = Marker]

matchCase: aWord using: modelWord

  "Make aWord conform to the case/capitalization of
   the model word."
   |word|
   (modelWord at: modelWord size) isUpperCase
     ifTrue: [^aWord asUpperCase].
   word := aWord asLowerCase.
   (modelWord at: 1) isUpperCase
     ifTrue:  [word at: 1 put: (word at: 1) asUpperCase].
   ^word

spell: aWord

  "Spell check a word: replace it if it has an entry
   in correctList, otherwise mark it if not in WordList.
   If the word is being replaced, make the case of the
   replacement word correspond to that of the input word."
   ^self
     matchCase:
       (correctList at: aWord trimBlanks asUpperCase
          ifAbsent:
            [(wordList includes: aWord trimBlanks asUpperCase)
              ifTrue:  [^aWord]
              ifFalse: [^aWord , (String with: Marker)]])
     using: aWord

words: wordFile

  "Add words from the specified file to the word list."
   self addWords: (Disk file: wordFile)
```

Listing 3. *(Continued from preceding page)*

We're getting hardnosed at Softway.

From now on MATIS™ is only $49^95!

(MATIS, the complete User Interface development tool has been selling for $150.)

Why the radical price cut?

We decided after looking over the competition that MATIS had so many advantages it should be made available to more programmers. We decided to compete aggressively so you could easily afford to have MATIS in **your** bag of tricks. We hear from MATIS users in the USA and France that it is a truly loveable product. Sooo...we're running this big ad to promote our new low price.

MATIS windows are beautiful.

Display any portion of any screens you create at any point in your program. Scroll in any direction manually with cursor keys...or automatically.

And the screens are HUMUNGEOUS!

MATIS screens can be just about as big as you want...up to 65,534 rows by 65,534 columns! The number of screens is only limited by available memory.

Print big MATIS screens directly.

One command sends your screens to your printer with no need to program special routines when your virtual screen is bigger than your terminal screen.

User input fields are a snap.

Creating fields for data entry is easy with no limit to size or number by screen. Request for input separately or in blocks.

Denis Moran
President, Softway, Inc.

Control your keyboard with MATIS.

It keeps track of keys that are pressed during the execution of your program and lets you assign specific functions to selected keys.

Control that screen too!

MATIS is extremely versatile and flexible when it comes to controlling lines, columns, fields, and text. They can be modified, transferred, displayed or moved with a single command. All video attributes are supported: color, reverse video, blinking...you name it, you got it.

Want an interactive screen builder?

You've got it with MATIS. It's called "MATPAGE"™ and it lets you create and modify any of your screens in an interactive mode.

MATIS adds over 70 routines to your program.

Written in Assembler, MATIS routines are fast and powerful giving your program improved efficiency and enhanced visual appeal, while they reduce its size and maintenance worries. And MATIS separates screen design from the core of your program.

MATIS is unique.

We don't think there's a single program that combines as many tools in one package as completely or as well as MATIS. It interfaces with Intepreted and Compiled BASIC (Microsoft), C (Lattice, Microsoft, Aztec), PASCAL (IBM, Microsoft) and ASSEMBLER. All you need is an IBM* PC/XT or true compatible under DOS, 128k or RAM, monochrome or color monitor.

You get an easy to follow no-frills manual and a 30-Day Money Back Guarantee.

Late News: MATIS/T™ for TURBO-PASCAL** only $29.95

An indispensable add-on at a dynamite price. What more can we say?

Denis Moran
Denis Moran

™ MATIS, MATIS/T, & MATPAGE are Trademarks of Softway, Inc.
*IBM is a Reg. Trademark of IBM Corp.
**Turbo Pascal is a Reg. Trademark of Borland International

Softway, Inc.

500 Sutter Street, Suite 222 CH
San Francisco, California 94102
(415) 397-4666

24-Hour Credit Card Orders By Phone:
1 (800) 227-2400 EXT 989 In California: 1(800)772-2666 EXT 989

Please ship the following at once. I understand there is a 30-day money back guarantee.

_____ Copies of MATIS at $49.95 plus $3 shpg. $ _____
_____ Copies of MATIS/T for TURBO PASCAL at $29.95 plus $2 shpg. $ _____
California residents, add 6½% sales tax $ _____
☐ I like to read specs, so send me a folder. Total $ _____

Name _____
Address (please no P.O. Box) _____
City/State/ZIP _____
Phone ( ) _____ Signature _____
Make payment by money order, check, or charge card ☐ VISA ☐ MASTER CARD
Number _____ Exp. _____

Distributed in Europe By: MICRO APPLICATION SOFTWARE • 147 Avenue Paul Doumer, 92500 RUEIL MALMAISON FRANCE, Tel (1) 732.92.54

CIRCLE 29 ON READER SERVICE CARD

BLISS Lures System Programmers

By Bruce Leverett

BLISS is a language designed for system programmers. This term is used loosely to denote two groups of people: those who write operating systems and those who read operating system programmer's manuals. BLISS is one of several languages developed between 1965 and 1975 for these people. It can be compared with C, a much better known language used by roughly the same kind of programmers.

BLISS originally was designed at Carnegie-Mellon University,[1] Pittsburgh, Pa., and has since been adopted by DEC for internal use.[2] At the time BLISS was designed, it was felt that system programmers had to be lured away from using assembly language by means of high-level languages with equal capabilities. Thus, BLISS offers much of the hands-on feel of assembly language programming with minimal built-in run-time support, while retaining the syntax of ALGOL-like languages. In addition, all BLISS compilers have been sophisticated optimizing compilers.

To the casual glance, a BLISS program looks much like an ALGOL-68, C, or Pascal program. Each of these languages has its own favorite punctuation and keywords, but the experienced programmer soon learns to discount the surface differences. Why, then, do we call BLISS exotic? It is not the syntax that is unusual but the semantics. BLISS brings along its own philosophy of system programming. One can learn a great deal about system programming just by studying this language. It is also instructive to compare BLISS with other languages or to compare the Carnegie-Mellon version with the subsequent DEC version. In this article I will describe a few of the highlights of the BLISS language design.

Distinguishing features

In BLISS, the name of a variable always refers to the address of the variable, never to its value. To get its value, you must use an operator, the dot (.). Since the main purpose of a variable is to hold a value, BLISS programs have a lot of dots. Here are some typical assignments in more conventional languages and their BLISS equivalents:

Pascal	C	BLISS
A := B	A = B	A = .B
A := B + C	A = B + C	A = .B + .C

The assignment *A = B* copies the address of B, not its value, into A. The equivalent assignment in the C language would be *A = &B.*

Why this departure from convention? The principal reason is to allow programmers to manipulate addresses easily and comfortably. In particular, the structure definition facility described later on could hardly be conceived of in a language with the usual semantics. Many languages don't allow a programmer to get at the addresses of variables at all. Those that allow the programmer to access these addresses have elaborate rules for specifying when the name of a variable denotes its *lvalue* (or address) or its *rvalue* (or value). In the wild free-for-all of pointers, integers, records, etc., involved in a complicated data structure access, elaborate rule systems can be useless or worse. But I am getting ahead of myself.

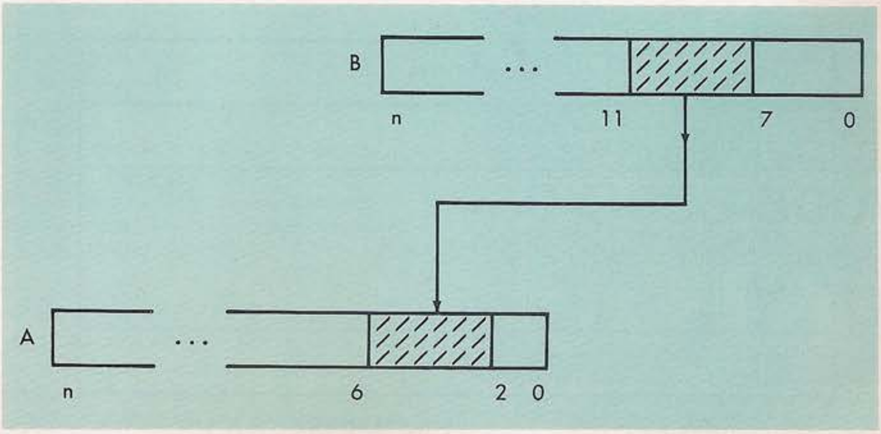
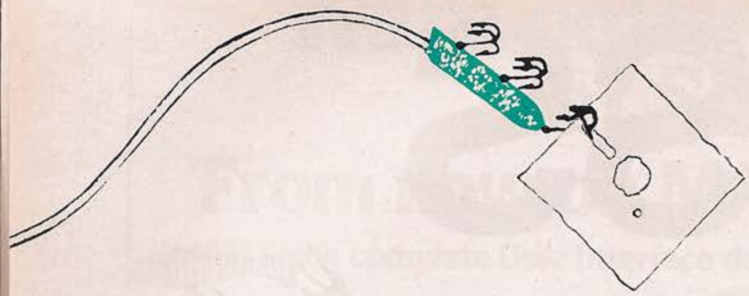


Figure 1.



Any contiguous subfield of bits in a word can act as an address. In other words, it can be assigned to or fetched from. For instance, the assignment:

```
A <2, 4> = .B <7, 4>
```

copies the value of a group of 4 bits within B into another group of 4 bits in A (Figure 1).

The exact rule is this: <*m*, *n*> is a group of *n* bits starting at bit *m*; bits are numbered from 0 up, with bit 0 being the least significant. For instance, on a DEC VAX-11, A <0,8> denotes the byte at address A. Many computers do not provide good hardware support for this kind of addressing. For instance, on the DEC PDP-11, the previously listed assignment would require quite a lot of shifting and masking, perhaps making copies of A or B. The choice of the "best" code sequence, from the point of view of a compiler, is not an easy one, depending as it does on which byte boundary each field is close to and whether or not the variables are in registers.

To the assembly language programmer, then, this feature is a significant abstraction. It allows him or her to play freely with the positions of fields within words without losing sleep over tricky modifications to long code sequences.

When system programmers write manuals and other documentation for each other, they often use diagrams like the one in Figure 2. This is a diagram of the layout of information in a block of 32-bit words. Each box represents a field; in this case, the *parms* field is an array. It may be that the size of this array is stored in the *size* field, so the program cannot access the *mung* field without first looking up that size. To add a further complication, it may be that some additional fields exist in a separate block pointed to by the field *gorp*, a kind of layout that is common when blocks share data with each other.

The possibilities available to the programmer who designs such a layout are limitless. In many languages, type definition facilities allow the programmer to specify some kinds of layouts using records or arrays, but a finicky programmer inevitably will be disappointed by his or her inability to get just the layout he or she has in mind. In BLISS, the programmer specifies an arbitrary algorithm that, given a base value (usually but not necessarily an address) and a field name, array index, or sequence of names and indexes, computes the address of the field to be accessed. This address can, of course, be a bit-field address, as described earlier.

Space does not permit me to show the syntax of these structure definitions, but suffice it to say that they are similar in appearance to subroutine definitions. Accesses to structures, no matter how bizarre the structure access algorithm, have a rather conventional appearance. For instance, supposing that F is the address of a block with the layout in Figure 2, the following accesses to the block might take place:

```
F [foo] = 3;
F [parms, .i] = .AnotherArray [.i];
X = .F [mung];
```

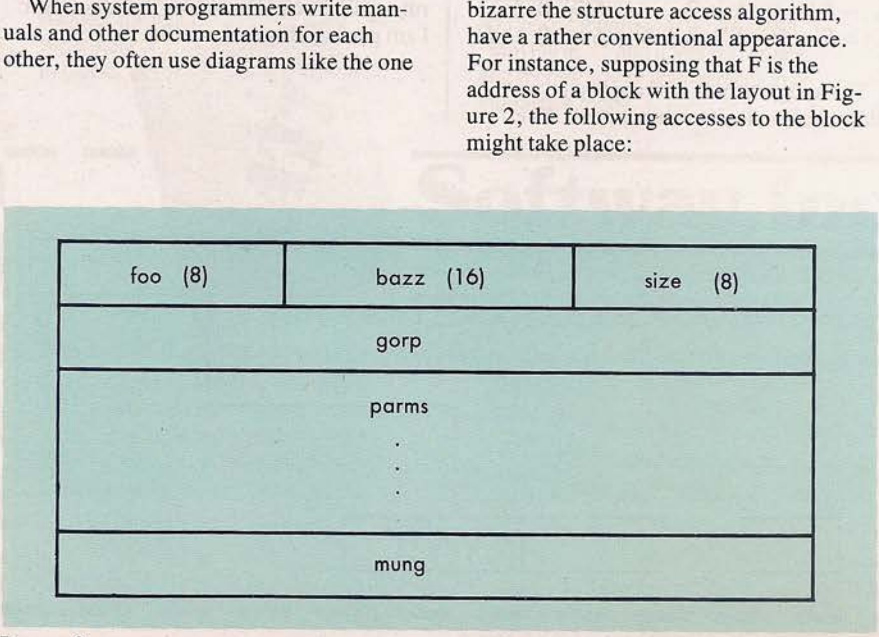


Figure 2.

The hidden computations behind these statements can be rather startling, for example, when looking up the *size* field to look up the *mung* field in the last statement. However, this example was chosen only to illustrate the generality of the structure definition feature. The programmer who does not use this generality does not pay for it; there are no hidden computations for conventional records, and for conventional arrays, the only hidden computation is scaling to access full-word elements on computers that are byte-addressed. System programmers can talk to their compiler just as freely and expressively as they talk to each other, without losing sleep over how it will understand or misunderstand their intentions.

I have described the three features of BLISS that distinguish it, in a truly instructive and useful way, from other languages used for system programming. The language has other unusual features, but as they are much less important, I will list them briefly.

- BLISS is an expression language with no distinction between expressions and statements; a statement is nothing more than an expression whose value is not used. The value computed by a block is defined as the value of the last of its statements (or, properly speaking, the last of its expressions). This is a minor convenience in that the programmer can freely add extra statements to the computation of any expression.
- BLISS does not have a *goto* construct. It has some constructs for exiting from loops and other types of expressions. Despite all the breath that has been wasted over this issue, programmers do not find this feature of BLISS to be very important in their lives, either for good or for ill.
- BLISS has powerful features for defining preloaded data. In the original Carnegie-Mellon BLISS, this was done with a general macro facility. DEC BLISS has more advanced features that allow the programmer to use the same structure accesses to define data that he or she would use if initializing it at run time.
- BLISS is typeless. No type checking is done at compile time. At the time BLISS

was designed, it was thought that in system programs, which were notorious for manipulating bit fields within pointers, having wild kinds of variant records, and other practices, type checking might be more of a nuisance than a help. Hindsight suggests that some kind of type checking would have been useful. BLISS programmers constantly make errors that cause integers to be used as pointers and vice versa, especially when they forget to use dots. They generally agree that compiler checking for this kind of error would be worthwhile.

BLISS, past and present

The original published description of BLISS is from 1971.[1] A compiler for BLISS for the PDP-10 was in use at least from 1970 at Carnegie-Mellon. A cross compiler running on the DEC PDP-10 and producing code for the DEC PDP-11 became available around 1972. This compiler was used heavily at Carnegie-Mellon and some other sites and is still in use today. After several years of positive experience with both compilers, DEC revised and expanded the language and implemented compilers for the new language for the PDP-10, PDP-11, and VAX-11, an effort that began in 1975 and was more or less complete in 1978. Many large systems have been written in BLISS, notably the Hydra multiprocessor operating system,[3] the Scribe document formatter,[4] and the BLISS compilers themselves.[5]

The designers of BLISS thought that for a high-level language to catch on among system programmers, it would have to provide many of the features that system programmers claimed to depend on in assembly language coding: close control over the type of code generated, especially the layout of storage; generation of optimal or near-optimal code; and access to all the unusual features of the underlying instruction set.

In retrospect, it seems that system programmers are not so finicky after all. The widespread use of C, a language that is much more conventional than BLISS and is not supported by optimizing compilers, is an object lesson. Though C is a reasonably well-designed language, programmers choose it not for its design but for what it provides access to (UNIX on various computers). Unique languages such as SNOBOL and LISP are attractive for their own sake. More commonplace languages, such as those that are ALGOL-like, are attractive mainly for their availability. In this respect, code optimization is as much a hindrance as a help. No native BLISS compiler for the DEC PDP-11 was ever developed because the design of the original BLISS compiler for the PDP-11, a cross compiler that ran on the PDP-10, was too large for porting to a 16-bit minicomputer.

In spite of these reservations, I can wholeheartedly recommend the design of BLISS. I have used BLISS and other languages for many years and most recently have helped construct an automated translation system to convert programs from BLISS to C. There is simply no language other than BLISS in which system programming is so natural and the compiler provides the programmer with so much useful abstraction while so cleanly staying out of the way. To the programmer who is dissatisfied with the design of the language or languages he or she uses from day to day, I say: study the design of BLISS if you want to know how it should have been done. ■

References

1. W.A. Wulf, D.B. Russell, and A.N. Habermann. "BLISS: A Language for Systems Programming." *CACM* 14, 12 (1971): 780-790.
2. R.F. Brender. *Generation of BLISSes.* Carnegie-Mellon Univ., Dept. of Computer Science, Technical Report CMU-CS-79-125, May 1979.
3. W.A. Wulf, R. Levin, and S.P. Harbison. *HYDRA/C.mmp: An Experimental Computer System.* McGraw-Hill (Advanced Computer Science Series), 1981.
4. B.K. Reid. *Scribe: A Document Specification Language and its Compiler.* Ph.D. dissertation, Carnegie-Mellon Univ., October 1980.
5. W.A. Wulf, R.K. Johnsson, C.B. Weinstock. S.O. Hobbs, and C.M. Geschke. *The Design of an Optimizing Compiler.* American Elsevier (Elsevier Computer Science Library), 1975.

Bruce Leverett has a Ph.D. in computer science from Carnegie-Mellon Univ., Pittsburgh, Pa. As a student, he was one of the implementers of BLISS-11. He is currently involved in research and development for Unilogic Ltd. and Lexene Corp.

SAVE $15.00

On October 1st, 1985, the price for figFORTH will be increased to $89.95 • Order NOW and you can have the complete figFORTH system for only $74.95

FORTH for the CP/M & TRS-80

operating system computer systems

figFORTH from SOTA Computing Systems Limited is rapidly becoming the FORTH of choice for both the novice and experienced FORTH programmer. Featuring a complete, accurate implementation of the figFORTH model, figFORTH from SOTA Computing Systems Limited also offers:

- full featured string handling
- floating point
- screen editor
- assembler
- beginner's tutorial
- comprehensive programmer's guide
- exhaustive reference manual
- unparalleled technical support
- source listings
- no licensing requirements
- no royalty arrangements
- unbeatable price

For the best implementation of FORTH that money can buy -- at a truly affordable price -- order figFORTH from SOTA Computing Systems Limited today!

ORDER FORM

Gentlemen: I want to save money!

☐Enclosed is my: ☐check ☐money-order for $74.95 (U.S. Funds).

☐Bill my: ☐VISA ☐Mastercard
I have indicated my card number and expiry date below:

Please rush me my copy of figFORTH by SOTA for:

☐CP/M Version 2.xx
☐CP/M Plus (Ver. 3.xx)
5 1/4" format only - please indicate computer type:

the TRS-80 Computer I have indicated:
☐Model I ☐Model 4
☐Model III ☐Model 4P

NAME: ____
STREET: ____
STATE: ____ ZIP: ____
CARD TYPE: ____ EXPIRY: ____
CARD NO: ____
SIGNATURE: ____

ORDER TODAY
213-1080 Broughton Street
Vancouver, British Columbia
Canada • V6G 2A8

MasterCard ORDER BY MAIL OR BY PHONE VISA®

(604)688-5009

State-of-the-Art since 1981

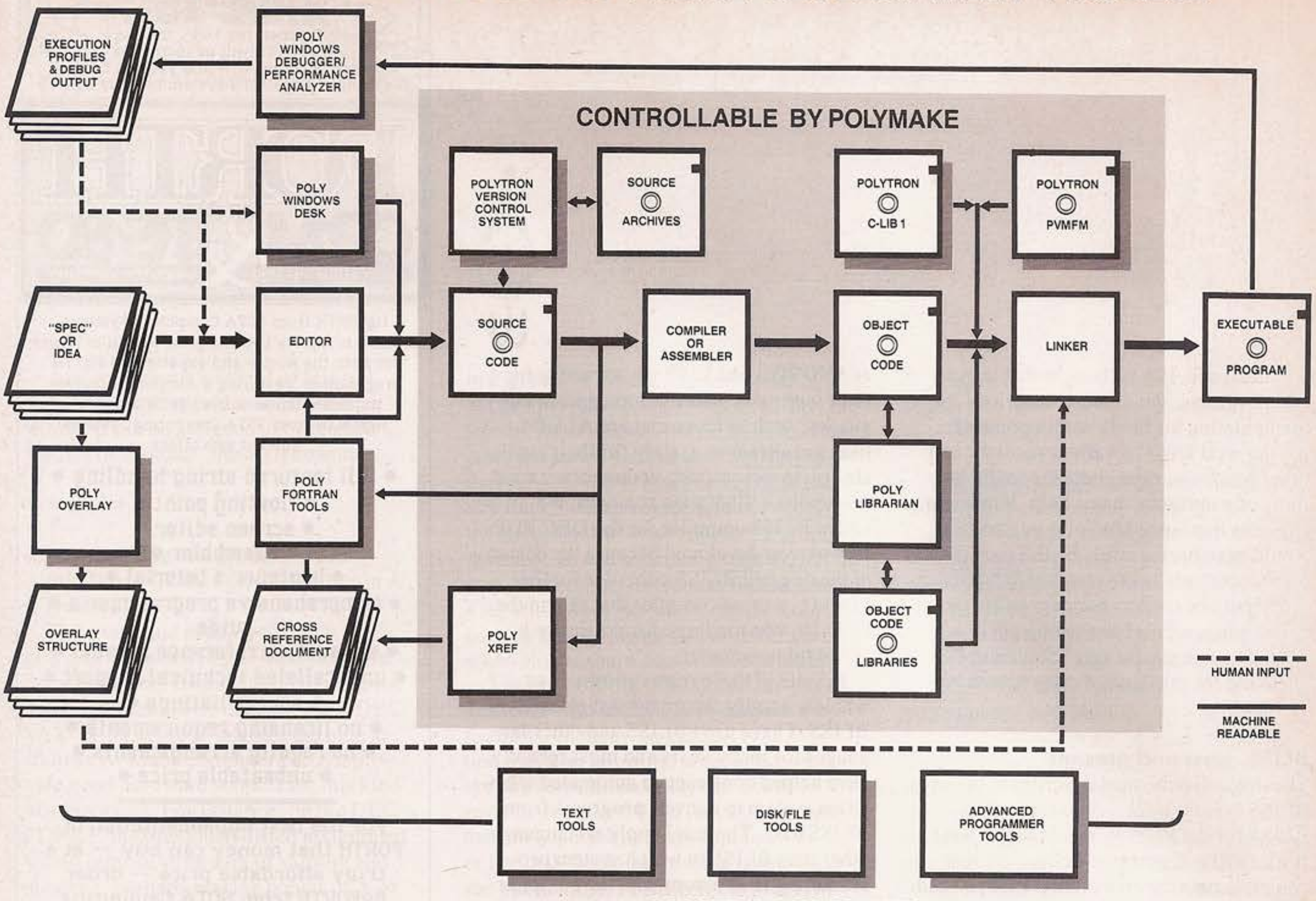
SOTA

Computing Systems Limited

TRS-80 is a registered trademark of Radio Shack
CP/M and CP/M Plus are registered trademarks of Digital Research

CIRCLE 36 ON READER SERVICE CARD

POLYTRON TOOLS IN THE SOFTWARE DEVELOPMENT PROCESS



POLYTRON TOOL KITS

Thousands of Professional Software Developers with demanding deadlines turn to high performance MS-DOS/PC-DOS tools from POLYTRON to boost their productivity. All the tools shown above are explained in detail in the **POLYTRON Programmer's Catalog.**

Network & Site Licenses Available.

PolyWindows Desk
The Modular, Expandable Desktop Organizer
100% memory-resident desktop organizer for professionals. Configurable so you can have one or more of the following functions available instantly: Hex/Decimal/Binary Calculator, Standard Calculator, ASCII Table, Multiple Document Text Editor, PolyKey for creating your own keyboard macros, Rolodex-type files, Calendar, Alarm Clock, Appointment Books, Autodialer, Grabber for cut-and-paste between applications. **Copy-Protected $49.95, Not Copy-Protected $84.95.**

PolyWindows Debugger/ Performance Analyzer
A Professional, Memory-Resident Software Development Tool
Instantly analyze and debug ANY program running on your PC anytime. Superior to the IBM Professional Debugger and easier to use. Supports multiple breakpoints, multiple display formats, memory searches, single-step execution, disassembly, hex arithmetic, and block moves. Tells you where your program is spending time and allows you to optimize performance. You can specify the memory ranges for analysis. Results are displayed graphically or in tables. **Includes PolyWindows Desk (Not Copy-Protected) all for only $149.** (Present PolyWindows users can receive trade-in discounts.)

POLYTRON Version Control System (PVCS)
Efficiently maintains revision history of source files for software project management. Maintains chronological, historical records of changes as "edit scripts" (reverse deltas) with full text of the latest "checked in" revision. Reconstructs any prior "revision" of any module, defines a "version" as specified revisions of various modules, supports branching from prior revisions, optional password protection. Friendly, unobtrusive user interface is based on a unique "syntax guide" concept with context-sensitive help. Designed for single or multiple programmer projects. A powerful tool with mainframe power. **Single-user license: $395.**

PolyXREF
The Multi-Lingual Cross Reference System
Generates Cross Reference listings across single or multiple source files of one or more languages. Tells you exactly where each variable and procedure is defined and everywhere it is referenced. **Package includes PolyXREF engine plus one source language module plus .CRF module: $129, additional source language modules [C, Pascal, Assembler or English]: $49 each. Complete package including all four language modules: $219.**

PolyLibrarian
The Object Module Library Manager
The premier library manager allows you to create, examine and manage libraries. Modules can be listed, added, replaced, extracted, and deleted. You can search for public or external names and even change them. Three operating modes can be intermixed for maximum flexibility. **PolyLibrarian I for Microsoft format libraries: $99, PolyLibrarian II for Microsoft and Intel format libraries: $149.**

PolyMake
The Intelligent Program Builder and Maintenance Tool
Automates the software maintenance process. Determines which files (programs, etc.) are out of date and invokes your compiler, linker, librarian and does whatever is necessary to bring your entire system up to date. Frees you from the need to remember which files depend on others and which files have been modified. Remembers the exact sequence of operations necessary to create a new revision. Completely automatic unlike competing products. Once you use it, you can't live without it. **Only $99.**

POLYTRON C Library 1
Over 65 High Performance Routines For Lattice C Compiler Users
These are routines for serious programmers that need Executive and I/O functions. Complete source code (mostly Assembler) included. **Only $99.**

POLYTRON Virtual Memory File Manager (PVMFM)
Provides efficient virtual file access and buffering for both text and binary files with automatic buffer-to-disk swapping. Reduces program access time and handles data structures larger than memory allows. PVMFM is a library of user-callable functions that may be linked with your applications code to provide virtual file management capability and random access file buffering. Compatible with Assembler and high-level languages. **Only $199.**

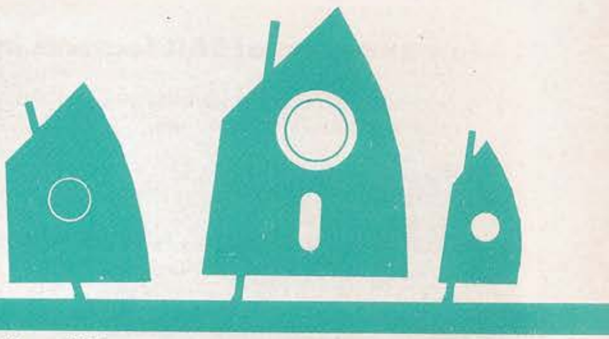
To Order Products Call
1-800-547-4000
Ask for Dept. No. 350
Foreign & Oregon orders call
(503) 684-3000
Send Checks, P.O.s To:
POLYTRON Corporation
P.O. Box 787 DS-350
Hillsboro, OR 97123
Add $5.00 Shipping To Total Order

To Order detailed Programmer's Catalog, send requests to address above for fast response (or circle reader service card number).

POLYTRON
High Quality Software Since 1982

CIRCLE 34 ON READER SERVICE CARD

PSAIL: SAIL to C



By Peter F. Lemkin

Wouldn't it be nice to be able to use dynamic text strings in a Pascal-like environment as we currently are able to do in BASIC? Well, it appears we can— plus more.

The SAIL language has had dynamic strings for years. The original SAIL compiler, developed between 1969 and 1976 at the Stanford University Artificial Intelligence Laboratory, Stanford, Calif., runs only on DEC-10 and DEC-20 computers. Unfortunately, most of SAIL was written in 36-bit dependent assembly language and is thus nonportable.

A portable version of SAIL, called PSAIL, is being constructed to run in most medium size C environments. This article will present PSAIL in the context of the original SAIL language.

What is SAIL? It is a robust ALGOL-dialect programming language with many useful extensions.[1,2,3] Because of its expressivity, SAIL lends itself to writing large, complex application and system programs. Its robust approach to type coercion makes it closer to a D.W.I.M (Do What I Mean) algorithmic language than other popular; block structured, strong type-checking languages such as Pascal, Modula-2 or Ada. On the other hand, C—which has looser type checking—does not have enough type coercion capability, especially in forcing arguments to the types expected in procedure calls. Without a lintlike syntax checker, writing large modular programs can be difficult.

Although not an ideal D.W.I.M. language by any means, SAIL comes closer to this approach than the other common block-structured languages mentioned. Of course, the price of more type coercion is that you can hang yourself more often.[4] Our experience, however, is that you learn to live with occasional pitfalls and take advantage of D.W.I.M. most of the time for greater overall productivity. Repeatedly SAIL has been selected in the National Institute of Health DECsystem-10 community as the implementation language of choice for large, complex systems.

Language overview

SAIL can be easily partitioned into several independent language subsets. This reduces what a new user has to learn to start programming quickly. These include: dynamic strings, dynamic arrays, dynamic records, macro expansion, conditional compilation, code inserted from other files, separate compilation of modules, bit manipulation, *if* and *case* statements and expressions, flexible I/O, powerful string scanning and conversion functions, LEAP (an associative data structure facility which includes dynamic data entities called items, datums, sets, lists, associations as well as associative search constructs—more on this later),[1,2,3] processes, contexts, interrupts, in-line assembly language, etc.

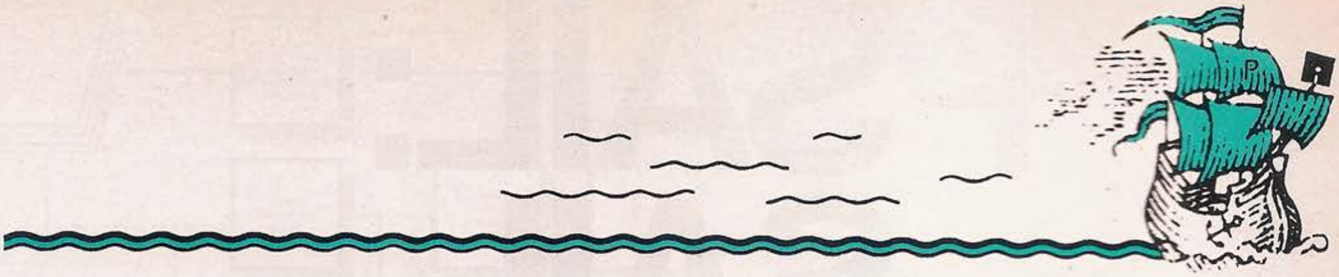
Note that complete exclusion of some of these sublanguages or features is easily achieved in actual programming practice. The most often used remainder is the core subset of SAIL, which captures its paradigm. This is relatively small—on the order of Pascal. For example, the PSAIL source code itself, written in SAIL, uses no records, LEAP, embedded assembly language, interrupts, processes, or context language constructs.

Table 1 illustrates some of the language elements with examples of SAIL fragments. Obviously, a much longer list would be required to illustrate the full language described in the *SAIL* reference manual.[1] In the examples, keywords appear in uppercase and user-defined symbols in lowercase. However, SAIL does not make alphabetic case distinctions as C does.

The PSAIL compiler project began as a result of the announced demise of the 36-bit/word DEC-10 and the increasing popularity of small UNIX systems. I had written a large data analysis system in SAIL called GELLAB,[5,6] which involved image processing and data base analysis for two-dimensional electrophoretic gels. To export this system, as well as other SAIL programs, the SAIL source code needed to be translated to some other language.

PSAIL was derived from an early program called SAITOC, written to translate GELLAB to C. Later, the goal of PSAIL was changed from that of a translator performing a partial translation to a full compiler capable of:

- Handling any SAIL program and flagging illegal SAIL syntax, unimplemented SAIL language elements, and C portability problems



A few examples of SAIL features implemented in PSAIL

1. Dynamic strings, string concatenation, substrings, string garbage collection, and some explicit string conversion functions.

```
STRING s, s1, s2;  INTEGER i, j, k;
s1  := "Adam";  s2 := "Eve";
s   :=s1 & s2;
i   := LOP(s); COMMENT extract 1st char of s, shorten s;
s1  := s & s1[i for 10] & " AND " & s2[k for j];
r   :=CVD("3."&"14159");  s := CVS(j+2**k);
s   := "The value of k is " & CVS(k) & " decimal and" &
    CVOS(k) & " octal";
```

2. Extensive macro facility including conditional compilation.

```
DEFINE # = "COMMENT ";  COMMENT # is short for COMMENT;
DEFINE dbug(x) = "PRINT(""x="",x,CRLF)";
IFC (tops10=10 AND nFiles<15)
      THENC ... ELSEC ... ENDC
```

3. Dynamic n-dimensional arrays with ragged (i.e. > 0 or < 0 bounds).

```
REAL ARRAY abc[-10:10], xyz[p:q];
STRING ARRAY  xyzzy[500:1000,-7:25,-10:-5];
```

4. Optional dynamic array bounds checking which may be toggled on/off.

```
SAFE REAL ARRAY x[1:10]
NOW!UNSAFE x; ...  # No bounds checking on 'x';
NOW!SAFE x; ...    # Resume bounds checking on 'x';
```

5. Nestable variable declarations inside of LABELED blocks. Note redefinition of x and y and that QUOTED block labels must match.

```
BOOLEAN z;
DO BEGIN "block one"
   INTEGER x, y;
      BEGIN "block two"
      REAL x, y;
       ... Compute z ...
      END "block two";
   END "block one"
UNTIL z;
```

6. Type coercions automatically generated where they are least likely to cause trouble. (Some Pascal supporters might disagree with this!)

```
STRING s;  INTEGER i;
i := s;           s := i + "A";
IF s = "Q" THEN ...
```

7. Separate compilation with INTERNAL/EXTERNAL storage modifiers and REQUIRED source code file modules with optional VERSION checking.

Table 1. *(Continued on following page)*

- Compiling itself so that PSAIL would be portable and programs could continue to be developed in a SAIL environment
- Running on a wide variety of systems.

At the NIH, we have had FORTRAN, ALGOL-60, and SAIL available. Over the past decade, a large body of SAIL code (greater than 200,000 lines) was written for some very large systems (GELLAB, about 70K lines, MLAB,[7] over 25K lines, and MATEXT,[8] over 35K lines). Other SAIL programs include BRIGHT, PUB, and the original TEX. Like C, SAIL supports modularity but with most of the type-checking capability of Pascal and similar languages. SAIL has often been used to build quickly large, reliable systems.

Studies have shown that the majority of algorithmic code, especially code involving numerical routines[9] and salvaging user libraries, could benefit from a high-level language translation rather than rewriting the code from scratch in a target language. It is then possible to translate verbatim much of the existing SAIL code to C using PSAIL in those cases where 36-bit dependencies are not a problem.

Using a high-level language translator preprocessor with a target system compiler causes some additional overhead. The arguments made for the FORTRAN preprocessor RATFOR (added portability, added language functionality, and user productivity) also hold for PSAIL since a SAIL environment is superior to C for program development.

By modularizing large SAIL programs into small files that are separately compiled, the additional translation cycle time should not be excessive. W. Teitelman[10] suggests that short compilation times are conducive to rapid interactive development. PSAIL should be fast enough to minimize this time. It will continue to improve as processor speeds and memory size increase.

Selecting C

To achieve portability over a wide variety of target computer systems, a portable target language is required. By writing a machine-independent, run-time library in the same target language, using machine-conditional, high-level language code where required, the feasibility of portability is greatly enhanced.

Assembly language was ruled out because it is not portable. It was decided to use a high-level language as the target language; thus the compiler is more a translator or transliterator between high-level languages than a conventional compiler which generates assembly language or machine code.

A number of popular languages, including C, Pascal, Ada, Modula-2, and MESA, were evaluated for their ability to express high- and low-level structures available in SAIL. A. Feuer and N.G. Gehani[11] give a valuable comparison of Ada, C, and Pascal, exposing their strengths and weaknesses. A further constraint we imposed was that the language should be commonly available and consistently implemented. C was selected as coming closest to meeting most of these goals.

Others have also selected C as a target language for translators. Recently there have been a number of high-level language translators including: S-TRAN for BASIC-to-C, a Pascal-to-C by J. Peterson, and FORTRIX for FORTRAN-to-C. Another advantage of a C high-level language translator is that the target code is just C code. Generated code can be used with the PSAIL run-time library or merged with other C programs, which are becoming increasing available.

PSAIL is based on the published language standards for SAIL[1] and for C.[12] Brian Kernighan and Dennis Ritchie's *The C Programming Language* is developing into the ANSI X3J11 standard.[13] There are several advantages in trying to stay within the existing language standards. Both have been well documented by these existing reference manuals.

Adhering to the standards helps ensure portability of much existing SAIL code and target C code. However, a price for standardization is not being able to polish some of the rough edges of SAIL with more modern language forms. A PSAIL language extension facility lets us optionally do this to some degree, giving us the best of both worlds.

Sublanguages

In general, adding additional language facilities missing from Pascal and C to an abstract, block-structured language increases the complexity of the language. It is generally understood that one price of

```
INTERNAL INTEGER p, d, q;     # in defining module;
EXTERNAL INTEGER p, d, q;     # in other modules;
EXTERNAL STRING PROCEDURE match(REFERENCE STRING s);
REQUIRE "boobah.sai" SOURCE!FILE;
REQUIRE "48.55" VERSION;
```

8. Records, checked record classes and record garbage collection.

```
RECORD!CLASS spot  (REAL angle, rad;
                     INTEGER area, xMom, yMom;
                     RECORD!POINTER (spot) next;
                    );
RECORD!POINTER (spot) rp1;
spot:xMom[rp1] := spot:rad[rp1]*COS(spot:angle[rp1]);
```

9. More natural Boolean relations as well as assignment embeddings.

```
STRING s; INTEGER ch;
s := "This string has 28 characters.";
WHILE s NEQ NULL DO
    If ("0" LEQ (ch:=LOP(s)) LEQ "9") OR
        (ch= ".") DO OUTSTR(ch);
    IF "Z" LEQ ch LEQ "A" THEN ch := ch-"A"+"a";
```

10. String scanning facility comparable to C's scanf() and scans().

11. Arrays which are dynamically defined and active in local blocks.

```
For i:= 1 STEP 10 UNTIL 101 DO
    BEGIN "Perform dynamic array allocation"
    REAL ARRAY vectorA[0:i], matrixB[0:2*i,0:i];
        . . . process the arrays . . .
    END "Perform dynamic array allocation";
```

12. Additional control statements: NEXT<label>, DONE<label>, CONTINUE<label> where <label> is optional.

```
DO BEGIN "outer loop of nonsense code"
     INTEGER x; REAL y, z;
     FOR x := 0 STEP 1 UNTIL 511 DO
        BEGIN "Loop x"
        FOR y := 25 STEP -(x+0.141) WHILE (x Leq 3001) DO
            BEGIN "Loop y"
            IF (z:=x+y)=123 THEN CONTINUE "Loop x";
            IF (z:=x-y)=1234 THEN DONE "Loop x";
            IF (z:=z+y)=321 THEN NEXT
                                ELSE NEXT "Loop x";
            END "Loop y";
        END "Loop x";
     END "outer loop of nonsense code"
UNTIL z<3.14;
```

Table 1. *(Continued from prrceding page)*

increasing the power of a language by extending its vocabulary and semantics is to increase the complexity perceived by the listener or writer in understanding and using the language. Computing history has shown that many programmers avoid the excessive complexity and size of languages such as PL/I and ALGOL-68. Even new languages such as Ada and MESA suffer from similar problems.

In addition, complex programming language solutions are more difficult to construct and port to other systems. I would suggest that it is this cost of portability that is partly responsible for the demise of these large languages' popularity.

So what can we do to get expressive power in a language at minimum cost? I suggest breaking the language into natural sublanguages so that programmers can use as much of the language as they want (or need)—and require the compiler enforce these language partitions.

We are experimenting in PSAIL with the hypothesis that a reduction in apparent language size is achieved by partitioning a large language into smaller disjoint sublanguages enforced by the compiler and held together by other connecting sublanguages. This should be reflected in reducing apparent complexity as observed by the user and would seem especially useful if some exotic aspects of the language are only needed in one or two modules.

Programmers normally think about their programming environment in terms of a language subset. A compiler should be able to enforce this thinking by having the programmer declare sublanguages either to exist or other sublanguages to be excluded. PSAIL currently contains the following four sublanguages and can handle more defined by the user:

- L0 = ALGOL-60 subset, strings, macros, etc.
- L1 = LEAP associative structure language
- L2 = Processes events and interrupts
- L3 = PSAIL extensions
- L4 = User definable

. . .

As a consequence of language partitioning:

- Less experienced users can work with a reduced language, which is easier to learn.
- Sophisticated users can expand the language to take advantage of advanced features.
- The PSAIL L3 extensions can be added. These currently include: generic procedures, array slices, the C ++, —, <opr>= operators, and embedded C code.
- Programmers may supply their own dynamic extensions to PSAIL (through *PSAIL!FORGET* and *PSAIL!DEFINE* compiler directives to modify the compiler by adding new keywords using existing parser capability as well as associating new run-time procedures). These extensions can become available to other users by specifying the new language definition in a REQUIRE file module. Some possible extensions might be to implement a concurrent LISP or add relational data base language extensions.

Portability

By assuming a consistent standard C environment such as ANSI X3J11, another hurdle to portability is overcome as target language code generators are easier to write. By divorcing the code generated by PSAIL from the much more machine- and operating system-dependent run-time library, we greatly facilitate the ease of porting SAIL source code between systems. The PSAIL run-time library consists of a number of *#include* type .h header files for C code files <sairun.c>, <pmath.c>, <gcrun.c>, <leaprun.c> and <procesrun.c>.

A potential disadvantage of this approach is that some run times could call <stdio>—effectively doing double interpretation. To avoid this, PSAIL run-time packages do their own file buffering and string handling rather than calling <stdio.c> repeatedly where major bottlenecks would appear.

These run-time libraries are also written in the same portable dialect of C that PSAIL generates. They use conditional C code, which is machine dependent only when required to handle machine-specific problems, whereas PSAIL generated code is completely machine independent. This method has been used for years in writing portable C code for different UNIX environments. One needs to compile the PSAIL run-time library only once after adjusting the <config.h> file to the specific system being ported.

PSAIL abstracts some of the code generation to a higher level than basic C code. For instance, whether the target system has a 32- or 16-bit integer is not important since PSAIL emits all integers as type *INTEGER*. This in turn will be defined differently in the only machine-specific file <config.h> for the two classes of machines, for example:

```
typedef int INTEGER; /* for 32-bit
                         word int */
```

or:

```
typedef long INTEGER; /* for 16-bit
                          word int */
```

Strings are another instance of this abstraction. PSAIL uses the C string pointer strategy to facilitate use of PSAIL strings with other C packages. The *STRING* declaration uses the C *typedef* of:

```
typedef char *STRING;
```

Using a compacting string garbage collector, PSAIL keeps track of all active string pointers using a run-time string pointer stack. Garbage collection is performed when the string allocator run-time *salloc()* runs out of space. For example, *salloc()* is called from the concatenation run-time procedure *catlist()* when it needs space the size of the strings to be concatenated. Strings may be as large as the memory available from the system. Another very useful SAIL feature is dynamically allocated arrays, which may have nonzero or negative lower bounds (sometimes called ragged arrays) and optional bounds checking.

The SAIL subset used in PSAIL was derived using several constraints. By restricting the language of PSAIL to a useful subset of full SAIL constrained by several factors, the resulting PSAIL language is a relatively robust portable language. These constraints include:

- A few subsets of the full SAIL specification are omitted at this time from PSAIL as few SAIL users in our programming community use them—for example, contexts. Although LEAP is translated, run-time code will not be written for L1 (or L2) in the first release of PSAIL. The open library scheme permits PSAIL users to redefine or write these initially missing run times in C.
- Features that are difficult to map to C or expensive to implement in terms of run-time efficiency are not currently implemented—for example, contexts. Similarly, PSAIL does not currently support nested procedures or global *GOTO*s, although code with warnings is produced.
- Features that take advantage of the DECsystem-10 specific instruction set but are not efficiently simulated on different architectures (for example, variable byte size operators for 36-bit words) are not currently implemented.
- There are also subtle data structure difference issues. In all of these cases, warning errors or comments are issued for the programmer. PSAIL's "warning" and "trash graphics" comments are used to catch these types of problems. For example, as the DECsystem-10 has a 36-bit word size, 36-bit arithmetic operations are also checked and when found, reported using "trash graphics."

PSAIL, a language translator as well as compiler, uses the philosophy of optional embedded C-style warning messages to warn about possibly illegal constructs. A summary appears at the end of compilation. Warnings are given both for SAIL source code syntax errors and target C code that would be nonportable due to limitations of C or in computation (for example, 36-bit operations). Programmers should use these warnings to edit their SAIL source code and try again. These forms include:

- "Trash graphics" are used to warn of possible problems. There is a continuum of warning levels to let you pick the message level you are comfortable to work with (Figure 1).
- Embedded warning C-style comments of the form:

```
/*WARNING*/ = check manually for
                possible semantic or
                syntax problem
/*UNDF*/ = Undefined symbol
/*N.P.*/ = Nonportable
/*N.I.*/ = Not implemented
/*LEAP*/ = LEAP sublanguage syntax
```

- Fatal SAIL syntax error messages that indicate what is wrong with the code, suggest what should be fixed (if it can figure it out), and point to the actual illegal SAIL source code.
- C portability tests are built into PSAIL. For example, you can optionally test for identifier name uniqueness to *n* characters. The proposed ANSI X3J11 standard has six-character external symbols, but other systems may have different lengths.

Although both SAIL and C are block-structured languages, simple one-to-one translation from SAIL to C is not always possible. For example, the following SAIL and C fragments are similar but not a simple 1:1 mapping:

```
SAIL: For i:=(a+b) Step -c Until d Do
    <SAIL statement>;
C: for (i=(a+b); i>=d; i-=c) <C
    statement>;
```

Therefore we perform simple translation where possible and elsewhere do recursive descent parsing and generate the nonlinear mappings where required. In some cases, where it is impractical to generate in-line C code, run-time procedure calls are generated instead.

By abstracting the target language, more attention can be paid to local optimization, taking advantage of the consistency of the target language and of C's power as a high-level assembler language. For example, the SAIL fragments:

```
INTEGER ARRAY a[1:10,1:20,1:30];
INTEGER b;
STRING s1;

a[i,j,k] := a[i,j,k] + b;
(s1[k For 1] = "Q")
```

are optimized to:

```
(a) a[i][j][k] += b;
     /* No dynamic arrays, C-style */
(b) *aAry(&a,i,j,k,3) += b;
     /* Dynamic arrays, no bounds
        check */
(c) *aChk(&a,i,j,k,3) += b;
     /* Dynamic arrays, bounds check*/
(d) (*(s1+k-1) == 'Q')
```

rather than the unoptimized C code, which includes redundant calculations and higher overhead with a run-time call:

```
         /****************************/
         /*<5>    /\          /\      */
         /*        /-------\        */
         /*       --|       |--      */
         /*      |  | o  o  |  |     */
         /*        -\-------/-      */
         /*       |     \/    |      */
         /*       +           +      */
         /* BEWARE!                  */
         /****************************/

/* WARNING! SHIFT OF > 31-BITS IS NOT PORTABLE */
or
/* WARNING! LOGICAL OPERATION > 31-BITS IS NOT PORTABLE */
```

Figure 1.

Current status of SAIL features implemented in PSAIL

SAIL language feature	Implemented?
'DEFINE' macros	Yes, with arrays and recursion
Conditional macro expressions	Yes, = = > #if or eval in PSAIL
FORC, WHILEC, etc.	Not yet, extended SAIL macro facility
REQUIRE file statements	Yes, map= = > #include, run-time pkgs
ALGOL-60-like declarations	Yes
Declaration type checking	Yes
Automatic type coercion	Yes
Nested procedure declarations	Yes, but may be nonportable
Nested variable declarations	Yes
Dynamic arrays	Yes, arymk(), aryfree(), aChk()
Positive ragged arrays	Yes, both C and dynamic arrays
Negative ragged arrays	Yes, currently only for dynamic arrays
Safe array bounds checking	Yes, C arrays, aAdr() for dynamic
ALGOL-60-like control statements	Yes
String concatenation	Yes, a&b&c = = > catlist(a,b,c,0)
String operators	Yes, LOP,sub-strings, EQU(), etc.
Infix to prefix operators	Yes, MAX, MIN, ABS, etc.
PRINT and OUTSTR	Yes, map = = > pprintf(. . .) in <sairun.c>
TOPS10 I/O run time	Yes, most are handled in <sairun.c>.
TOPS20/TENEX I/O run time	Only those which are TOPS10 compatible
In-line assembly code	Yes, mapped to #asm/#endasm
RECORDS	Yes, mapped to struct's, no ralloc() yet
LEAP sublanguage	Most, maps through to <leaprun.c>
PROCESSES, EVENTS, INTERRUPTS	Most, maps through to <procesrun.c>
DEC-10 byte pointers in SAIL	Yes, Map= = >procedure calls, /*N.P.*/
CONTEXTS AI sublanguage	Not yet
<sairun.c>	Yes, written—not debugged
<gcrun.c>	Yes, written—not debugged
<pmath.c>	Yes, written—not debugged
<leaprun.c>	Not yet, not written, calls generated
<procesrun.c>	Not yet, not written, calls generated
STRING garbage collection	Yes, <gcrun.c> written—not debugged
RECORD garbage collection	Not yet, rec_gc() will mimic str_gc()
Procedure profiling	Yes, different from SAIL—not debugged
BAIL dynamic debugger	Not yet (maybe never . . .)
36-bit dependent code	Forget it! Recode your program!!!

Table 2.

```
(a) a[i][j][k] = a[i][j][k] + b;
(b) *aAdr(&a,i,j,k,3) =
      *aAdr(&a,i,j,k,3) + b;
(c) *aAdr(&a,i,j,k,3) =
      *aAdr(&a,i,j,k,3) + b;
(d) (*subsr(s1,k,1) = = 'Q')
```

The generation of these different array access forms *(a)-(c)* is controlled by *COMPILER!SWITCHES /CHECK* or / */NOCHECK, SAFE, NOW!SAFE, NOW!UNSAFE* compiler directives.

Status of PSAIL implementation

Because PSAIL can handle most of SAIL and has an extensive error and warning facility, the compiler is relatively large—certainly more than 64K bytes. No attempt is being made to try and squeeze it into today's toy computers since real machines with lots of memory and large disks are becoming increasingly available (at toy machine prices).

The first machine selected for export will be DEC's microVAX, then 68000-, 32016-, or 80286-class machines. Any reasonably sized machine with a decent C compiler is a likely candidate for porting PSAIL. It may also be possible to fit an optimized PSAIL into the limited memory 8088-class machines.

A SAIL code validation test suite and large number of actual working source code programs were and are being used as a compiler construction and debugging tool. These programs represent radically different styles and requirements in areas of numerical analysis, string processing, and data base analysis. Working program sources are extremely useful for finding subtle compiler bugs because, as has often been stated, no compiler is completely bug-free.

Table 2 lists the features currently implemented. It is the author's intention to place PSAIL in the public domain when it is released and to make it available on various bulletin boards and to user groups. (Watch for a *COMPUTER LANGUAGE* Users Group and Bulletin Board Service announcement of PSAIL's availability.) For those interested in working with this emerging, portable SAIL environment, it should be available later this year.

References

1. Reiser, J.F. *SAIL*. Stanford Artificial Intelligence Lab memo AIM-289 or Computer Science Report #STAN-CS-76-574 (1976). Also available as #AD-A045-102 from National Technical Information Service, Dept. Commerce, Springfield, Va. 22161. Microfiche $4.50 and paper $17.50.
2. Feldman, J.A., and P.D. Rovner. "An ALGOL-based Associative Language." *CACM* 12(8) (1969): 439-449.
3. Bobrow, D.B., and B. Raphael. "New Programming Languages for AI Research." *Computing Surveys* 6(3) (1974): 153-174.
4. Geschke, C.M., J.H. Morris, and E.H. Satterthwaite. "Early Experience with MESA." *CACM* 20(8) (1977): 540-553.
5. Lemkin, P., and L. Lipkin. "GELLAB: A Computer System for 2D Gel Electrophoresis Analysis." *Computers in Biomedical Research* 4 (1981): Part I, 272-297; Part II, 355-380; Part III, 407-446.
6. Lemkin, P.F., and L.E. Lipkin. "Database Techniques for Two-Dimensional Electrophoretic Gel Analysis." In *Computing in Biological Science,* edited by M. Geisow and A. Barrett, 181-231. Amsterdam: Elsevier North Holland. [ISBN-0-444-80435-8]
7. Knott, G. D. "MLAB—A Mathematical Modeling Tool." *Computer Programs in Biomedicine* 10 (1979): 271-280.
8. Lipkin, J. and B.S. Lipkin. "Data Base Development and Analysis for the Social Historian." *Computers in the Humanities.* 12 (1978): 113-125.
9. Freak, R.A. "A FORTRAN to Pascal Translator" *Software Practice and Experience* 11 (1981): 717-732.
10. Teitelman, W. "A Tour through Cedar." *IEEE Software* 1(2) (1984): 44-73.
11. Feuer, A., and N.G. Gehani. *Comparing and Assessing Programming Languages: Ada, C, Pascal.* Englewood Cliffs, N.J.: Prentice-Hall, 1984. [ISBN-0-13-154857-3]
12. Kernighan, B.W., and D.M. Ritchie. *The C Programming Language.* Englewood Cliffs, N.J.: Prentice-Hall, 1978. [ISBN-0-13-11-163-3]
13. Plum, T. "A Scorecard for Draft ANSI C." *The C Journal* 1(1) (1985): 8-13.

Peter Lemkin is a computer scientist at the National Cancer Institute of the National Institutes of Health in Bethesda, Md. He applies image processing and data base techniques to biomedical problems.

How to go from UNIX to DOS without compromising your standards.

It's easy. Just get an industry standard file access method that works on both.

C-ISAM™ from RDS.

It's been the UNIX™ standard for years (used in more UNIX languages and programs than any other access method), and it's fast becoming the standard for DOS. Why?

Because of the way it works. Its B+ Tree indexing structure offers unlimited indexes. There's also automatic or manual record locking and optional transaction audit trails. Plus index compression to save disk space and cut access times.

How can we be so sure C-ISAM works so well?

We use it ourselves. It's a part of INFORMIX,® INFORMIX-SQL and File-it!,™ our best selling database management programs.

For an information packet, call (415) 424-1300. Or write RDS, 2471 East Bayshore Road, Palo Alto, CA 94303.

You'll see why anything less than C-ISAM is just a compromise.

RDS™

RELATIONAL DATABASE SYSTEMS, INC.

© 1985, Relational Database Systems, Inc. UNIX is a trademark of AT&T Bell Laboratories. INFORMIX is a registered trademark and RDS, C-ISAM and File-It! are trademarks of Relational Database Systems, Inc.

CIRCLE 77 ON READER SERVICE CARD

The #1 Language Spotlights the Fast

C Seminar/Workshop

September 16-18
Cambridge, Mass

Sponsored by the creators of COMPUTER LANGUAGE

The C Seminar/Workshop will be an exciting forum on the latest technical innovations and C language developments. Best of all, you'll experience a practical, hands-on approach in small workshop sessions. Take full advantage of the powerful C language by attending the first COMPUTER LANGUAGE Seminar/Workshop.

The Seminar/Workshop will focus on the practical techniques and the latest developments you can apply in your everyday work. You'll learn from the foremost authorities about:

- **Writing code for machine portability**
- **Using C functions and declarations effectively**
- **Program development utilities**
- **Code readability and organization**
- **C statements, data types, and operators**
- **Code optimization techniques**

You won't be just an observer at this event. Lively discussions will take place in small workshop groups. You can select the specific workshops you're most interested in.

The C seminar speakers and workshop leaders represent a virtual *Who's Who* of the C programming community. Only COMPUTER LANGUAGE, the programming language expert's magazine, could draw together these C programming authorities. They represent a wide spectrum of expertise in every significant area of C.

The cost for this comprehensive 2½-day event is only $695.

Here are just a few of the speakers and workshop leaders:

JIM BRODIE: Overview of the ANSI Standardization Effort
RON CAIN: Programming Robotics in C
JIM HENDRIX: Network Data Base Theory and Small-C
HEINZ LYCKLAMA: Unix Perspective on C
P.J. PLAUGER: Programming Style and C
TOM PLUM: Efficiency of C Programs
JACK PURDUM: C Benchmark Methodologies
LARRY ROSLER: Language Standardization Issues
BOB VARNEY: C Development on the Macintosh
ROBERT WARD: Structured Methods of Debugging C
LEOR ZOLMAN: Porting C Programs between Operating Systems

WORKSHOPS

Seminar participants will be able to choose four topics from this list (subject to change based on availability):

- **Efficient Code Generation**
- **Philosophy and Methodology of Benchmarks**
- **ANSI Standards: Questions & Answers**
- **Code Readability and Organization**
- **Asynchronous Communications**
- **Cross Compilers**
- **Writing Extentions to C**
- **Programming Robotics in C**
- **C/UNIX System Subroutine Interfaces**
- **Porting C between CP/M, MS-DOS, and UNIX**
- **Interpreters in a Development Environment**
- **Programming for Portability**
- **Debugging Techniques**
- **Network Data Base Theory and Small-C**
- **Object-File Formats for UNIX Systems**
- **C Development on the Macintosh**
- **C Benchmark Methodologies**

Magazine Growing C Language

REGISTRATION INFORMATION

TUITION: The $695 course fee includes all lectures, six workshops, all course materials, proceedings, and lunch on days one and two.

LOCATION: Sheraton-Commander Hotel, Conference Rooms, 16 Garden St., Harvard Square, Cambridge, Massachusetts 02238 (617) 547-4800. Cambridge is considered by many to be one of the most cosmopolitan intersections of individuals and ideas with its two major residents, Harvard and MIT. What better place to hold a landmark seminar on the future of C!

REGISTRATION: To register simply call:

415-957-9353

OR detach registration form and mail to:

CL Publications Inc.
C Seminar
131 Townsend St.
San Francisco, CA 94107

Limited space is available. Early registration is recommended. Acceptance will be confirmed by mail. Only paid enrollments can be confirmed.

DISCOUNTS: Three or more enrollments from the same firm qualify for a $100 discount off each enrollment. Please submit multiple registrations and multiple payments together.

HOTEL ACCOMMODATIONS: A limited number of rooms is reserved at the Sheraton-Commander Hotel, where the seminar will be held. Contact the hotel reservations desk directly at (617) 547-4800 as soon as possible. For preferred rates please mention you are attending the CL Publications C Seminar.

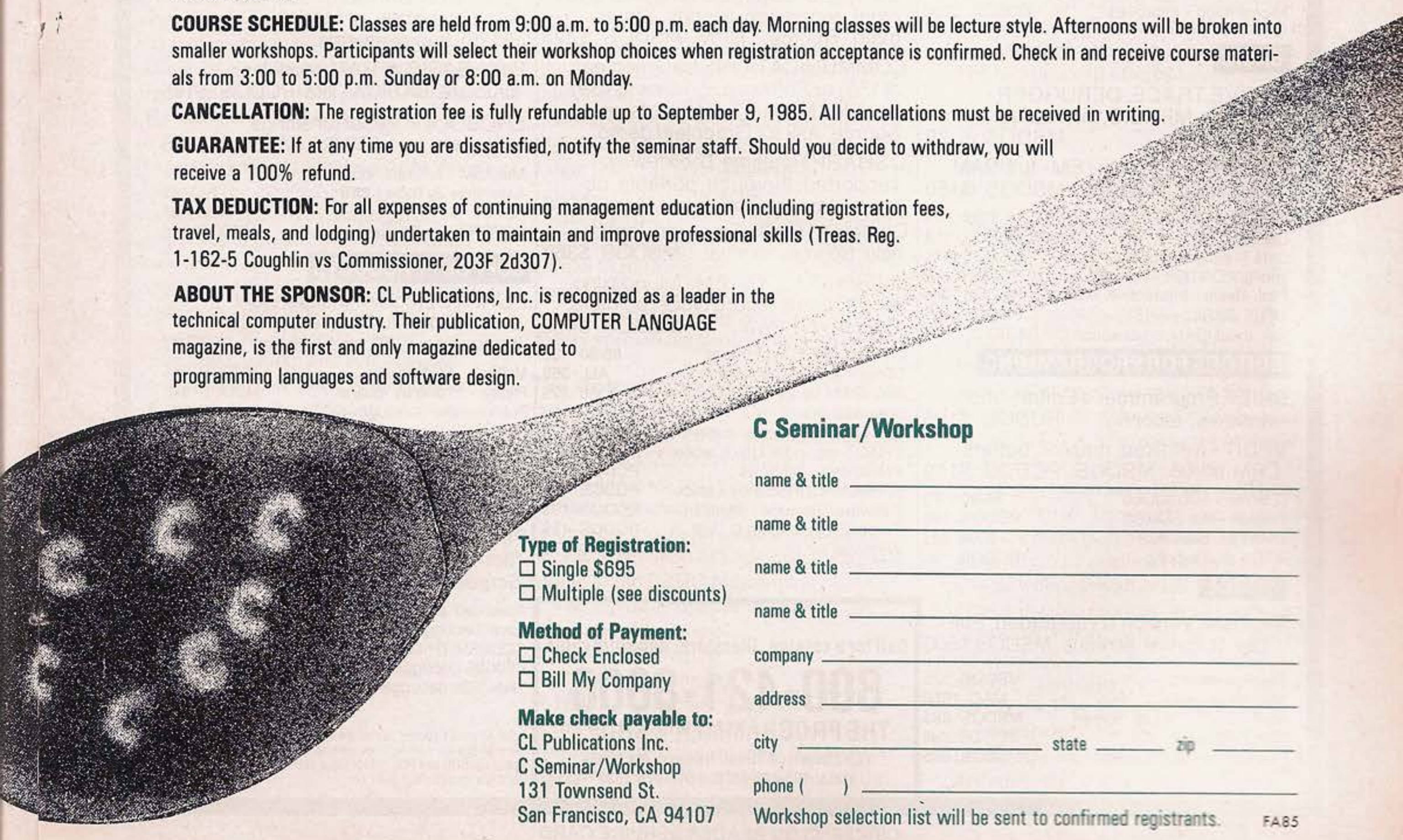
COURSE SCHEDULE: Classes are held from 9:00 a.m. to 5:00 p.m. each day. Morning classes will be lecture style. Afternoons will be broken into smaller workshops. Participants will select their workshop choices when registration acceptance is confirmed. Check in and receive course materials from 3:00 to 5:00 p.m. Sunday or 8:00 a.m. on Monday.

CANCELLATION: The registration fee is fully refundable up to September 9, 1985. All cancellations must be received in writing.

GUARANTEE: If at any time you are dissatisfied, notify the seminar staff. Should you decide to withdraw, you will receive a 100% refund.

TAX DEDUCTION: For all expenses of continuing management education (including registration fees, travel, meals, and lodging) undertaken to maintain and improve professional skills (Treas. Reg. 1-162-5 Coughlin vs Commissioner, 203F 2d307).

ABOUT THE SPONSOR: CL Publications, Inc. is recognized as a leader in the technical computer industry. Their publication, COMPUTER LANGUAGE magazine, is the first and only magazine dedicated to programming languages and software design.



Type of Registration:
☐ Single $695
☐ Multiple (see discounts)

Method of Payment:
☐ Check Enclosed
☐ Bill My Company

Make check payable to:
CL Publications Inc.
C Seminar/Workshop
131 Townsend St.
San Francisco, CA 94107

C Seminar/Workshop

name & title ____________________
name & title ____________________
name & title ____________________
name & title ____________________
company ____________________
address ____________________
city __________ state _____ zip _____
phone (   ) ____________________

Workshop selection list will be sent to confirmed registrants. FA85

CIRCLE 97 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature free
- Dealer's Inquire
- Newsletter
- Rush Order
- Over 700 products
- BULLETIN BOARD - 7 PM to 7 AM 617-826-4086

Free Literature - Compare Products

Evaluate products Compare competitors Learn about new alternatives. One *free call* brings information on just about any programming need. Ask for any ***"Packet"*** or "Addon Packet" ☐ ADA, Modula ☐ "AI" ☐ BASIC ☐ "C" ☐ COBOL ☐ Editors ☐ FORTH ☐ FORTRAN ☐ PASCAL ☐ UNIX/PC or ☐ Debuggers, Linkers, etc.

RECENT DISCOVERIES

PC LINT - full C program checking and big, small model. C86, Lattice. MSDOS $ 95

ARTIFICIAL INTELLIGENCE

ARITY/PROLOG-full, debug, to ASM&C, 16 Meg use, windows, strings. With compiler $1950. MSDOS $495

ExpertEASE - Expert system tool. Develop by describing examples of how you decide. PCDOS $625

Expert LISP - Interpreter: Common LISP syntax, lexical scoping, toolbox, graphics. Compiler. 512K MAC $465

EXSYS - Expert System building tool. Full RAM, Probability. Why, serious, files PCDOS $275

GC LISP - "COMMON LISP", Help. tutorial, co-routines, compiled functions, thorough. PCDOS Call

M Prolog - full, rich, separate work spaces. MSDOS $725

PROLOG-86 - Learn fast. Standard, tutorials, samples of Natural Language. Exp. Sys. MSDOS $125

TLC LISP - "LISP-machine"-like. all RAM, classes, turtle graphics 8087 CPM-86. MSDOS $235

WALTZ LISP - "FRANZ LISP" - like, 611 digits, debugger, large programs. CPM80 MSDOS $159

MicroProlog - improved	MSDOS	$235

BASIC

ACTIVE TRACE, DEBUGGER - BASICA, MBASIC, interactive, well liked MSDOS $ 79

CADSAM FILE SYSTEM - full ISAM in MBASIC source. MSDOS $150

BASCOM-86 - Microsoft	8086	279
CB-86 - DRI	CPM86, MSDOS	419
Data Manager - full source	MSDOS	325
InfoREPORTER - multiple	PCDOS	115
Prof. Basic - Interactive, debug	PCDOS	89
TRUE BASIC - ANSI	PCDOS	125

Ask about ISAM, other addons for BASIC

EDITORS FOR PROGRAMMING

BRIEF Programmer's Editor - undo, windows, reconfig. PCDOS $195

VEDIT - well liked, macros, buffers, CPM-80-86. MSDOS. PCDOS $119

C Screen with source	86/80	75
Epsilon - like EMACS	PCDOS	195
PMATE - powerful	8086	185
XTC - multitasking	PCDOS	95

COBOL

Microsoft Version II - upgraded. Full Lev. II, native, screens. MSDOS $500

Dig Res-decent	MSDOS	525
Macintosh COBOL - Full.	MAC	1850
MBP - Lev II, native, screen	MSDOS	885
MicroFocus Prof.-full	PCDOS	call
Ryan McFarland-portable	MSDOS	695

C LANGUAGE

C-terp Interpreter by Gimpel, full K&R, .OBJ and ASM interface. 8087 MSDOS $275

INSTANT C - Interactive development - Edit. Source Debug, run. Edit to Run - 3 Secs. MSDOS $445

"INTRODUCING C" - Interactive C to learn fast. 500 page tutorial. examples, graphics PCDOS $ 95

Wizard C - Lattice C compatible, full sys. III syntax, lint included, fast, lib. source. MSDOS $450

MSDOS C86-8087, reliable	call
Lattice C - the standard	call
Microsoft C 3.0 - new	279
Williams - debugger, fast	call
CPM80 - EcoPlus C-faster, SLR	275
BDS C - solid value	125

MEGAMAX C - native Macintosh has fast compile, tight code, K&R. toolkit, .OBJ, DisASM MAC $275

MACINTOSH Hippo II	375
Consulair's MAC C, toolkit	395

Compare. evaluate. consider other Cs

C ADDONS

APPLICATION TOOLKIT by Shaw - Complete: ISAM, Screen, Overlay mgnt, report gen. Strings, String math. Source. CPM, MSDOS $395

COMMUNICATIONS by Greenleaf ($159) or Software horizons ($139) includes Modem7, interrupts, etc. Source. Ask for Greenleaf demo.

C SHARP Realtime Toolkit-well supported, thorough, portable, objects, state sys. Source MANY $600

CIndex + -full B+Tree, vari. length field. Source, no royal. MSDOS $369

dbVista FILE SYSTEM - full indexing, plus optional record types, pointers. Source, no royal. MSDOS $450

CHelper: DIFF, xref, more	86/80	135
CTree - source, no royal	ALL	369
dBC ISAM by Lattice	8086	229
Greenleaf-200 +	MSDOS	159
OTHER: C Utilities by Essential	MSDOS	129
PHACT-up under UNIX, addons	MSDOS	225
ProScreen - windows	PCDOS	275
SCREEN: CURSES by Lattice	PCDOS	125
Software Horizons - Blocks I	PCDOS	139
Turbo V - Greenleaf C, fast	PCDOS	159
Windows for C	MSDOS	175

Call for a catalog, literature, and *solid* value

800-421-8006

THE PROGRAMMER'S SHOP™

128-L Rockland Street, Hanover, MA 02339
Mass: 800-442-8070 or 617-826-7531 8517

FORTRAN LANGUAGE

MacFORTRAN - full '77, '66 option. toolbox, debugger, 128K or 512K. ASM-out option MAC $375

RM/Fortran - Full '77. BIG ARRAYS. 8087, optimize, back trace, debug. MSDOS $525

Ask about Microsoft, Supersoft, others.

MS FORTRAN-86 - Improved.	MSDOS	239
DR Fortran-86 - full '77	8086	249
PolyFORTRAN-XREF, Xtract	PCDOS	165

LANGUAGE LIBRARIES

MultiHALO Graphics-Multiple video boards, printers, rich. Animation, engineering business. Any MS language, Lattice C86 $195, for Turbo $95.

Screen Sculptor - slick, thorough, fast, BASIC, PASCAL. PCDOS $115

GRAPHMATIC - 3D, FTN, PAS	PCDOS	125
File MGNT: BTrieve - all lang.	MSDOS	215
Micro: SubMATH - FORTRAN full	86/80	250
MetaWINDOW - icons, cup	PCDOS	139
PANEL - many lang. term	MSDOS	249

OTHER LANGUAGES

ASSEMBLER-ask about Turbo ASM ($95), ED/ASM ($95) - both are fast, compatible. or MASM ($125), improvements.

BetterBASIC all RAM, modules. structure. BASICA - like PCDOS $185

SNOBOL4 + -great for strings, patterns. CPM86, MSDOS $ 85

MacASM - full, fast, tools	MAC	115
Assembler & Tools - DRI	8086	159
PL1-86	8086	495
PCFORTH - well liked	MSDOS	95

SUPPORT PRODUCTS

PLINK - a program-independent overlay linker to 32 levels for all MS languages. C86 and Lattice. $315

Multilink - Multitasking	PCDOS	265
Pfinish - Profile by routine	MSDOS	345
Polylibrarian - thorough	MSDOS	95
PolyMAKE	PCDOS	95
ZAP Communications-VT100. TEK 4010, full xfer	PCDOS	65

DEBUGGERS

Periscope Debugger - load after "bombs", symbolic, "Reset Box", 2 Screen, own 16K. PCDOS $279

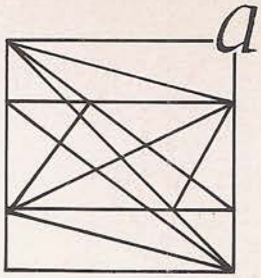
Advanced Trace 86 Symbolic	PCDOS	149
Atron Debugger for Lattice	PCDOS	395
CODESMITH-86 - debug	PCDOS	129
PFIX-86 Debugger	MSDOS	169
TRACE86 debugger ASM	MSDOS	115

Note: All prices subject to change without notice
Mention this ad. Some prices are specials.
Ask about COD and POs. All format's available
UNIX is a trademark of Bell Labs

CIRCLE 35 ON READER SERVICE CARD

Part II: The ERGO Logic Editor

Logic at a Glance



By Jim McCarthy

In this installment of our description of the ERGO Logic Kit (see Part I in COMPUTER LANGUAGE, *July 1985), we will focus our attention on those parts of the kit that provide facilities for the creation and editing of decision tables, with an eye toward compilation. We will discuss the features available, why they exist, and what some of the compromises involved in implementing them were.*

It is reasonable to expect that these topics will be of most interest to programmers. Therefore the tenor of this article (including the examples), will be slanted in a technical direction.

However, those of you who may be more interested in the expert system class of application engendered by the ERGO Logic Kit would probably do well to bear with us, as the ERGO Logic Interpreter (described in detail in next month's installment) uses the same algorithms and techniques as the ERGO Logic Compiler for decision table parsing and rule management. Also, the same Logic Editor is used whether you're creating source code or executable tables. And finally, the rationale and philosophy of the ERGO Logic Kit are described in some detail, and these are applicable across all elements of the kit.

For information on how to acquire your version of the ERGO Logic Kit (at a nominal cost) see page 4.

The ERGO Logic Editor differs from general purpose electronic text editors and word processing programs in that it will be useful only when the class of input is logical in nature. No provision has been made for general text entry, for example. Also, ERGO makes no claims to be in that class of integrated software that includes multiple, generalized applications; however, this system is meant to be a multifunctional application within the domain of logic and logic programming, and the Logic Editor is at the heart of the manifold functions ERGO provides.

Because the Logic Editor was designed to work within the confines of a particular format used to encode and manipulate logic known as a decision table (see last month's article), many features of a highly specialized nature are provided.

The status display

Refer now to Figure 1. This is a representation of a Logic Editor screen. (Note: the C1 through C7 and A1 through A7 label entries and code will be described later.) Defining the elements of this screen will be the best way to give you a glimpse of the ERGO method of logic processing and a view of the ERGO philosophy.

The top three lines of the display are always used for one of:

- Status information
- Command menus
- Helpful information.

In Figure 1, the top three lines are devoted to the standard table status display. When this display is presented, the ERGO Logic Editor is in ready mode, that is, condition and action labels, stubs, and entries may be keyed in.

The top left element on the status display is the table name. This is merely any arbitrary, legal file name. Permitted characteristics of the file name are, of course, dependent on the host operating system's expectations.

On the right side of the screen, top line, is a location indicator. This describes the editorial spot where the cursor resides. The location will always be one of:

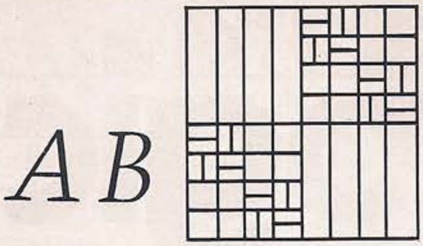
LABEL	r
STUB	r
MATRIX	r
ENTRY	r:c
ACT LABEL	r
ACT STUB	r
ACT ENTRY	r:c

where *r* equals the current row and *c* equals the current column. The reason a location indicator is required is that the Logic Editor permits you to scroll around a large, virtual table, and it can be reassuring to know where you are in the table and that the editor knows, too.

I suspect that this indicator will be only of modest use. This is because in decision table construction, as in software design generally, a key value is to keep the units you're working with small. As proficiency in table fabrication and parsing grows, the cursor will always be precisely where it appears to be in a table matrix, and the "virtuality" provided by the editor may well be a feature that atrophies.

Nonetheless, the feature seems appropriate: it is there when you need it as a beginner, and it is in line with probable

Illustrations: Barbara Luck



user expectations, given the extensive use of numerical and data-based spreadsheets that encourage you to scroll hither and yon.

Below the table name is the rule name indicator. This is useful for a variety of reasons. If you choose to, you can name each of your rules (entry columns). Then, as you move through the rules, creating, editing or otherwise relating to them, the mnemonics you assigned to the rules flash before you at this spot.

This can be tremendously important, because scrutinizing a column of y/n or a/b/c type entries does require a moment's (or several moments') reflection before the meaning of that column crystallizes in your mind. Much delay can be avoided with rule names. Also, the rule names become a sort of accountability feature: the name specifies in a natural language what the rule is, and you can balance that natural language specification against the more abstract depiction of the same rule in a logically encoded format in the rule column itself. That is, the rule name should be equal to the rule entries—they're merely two different ways of giving form to the same substance.

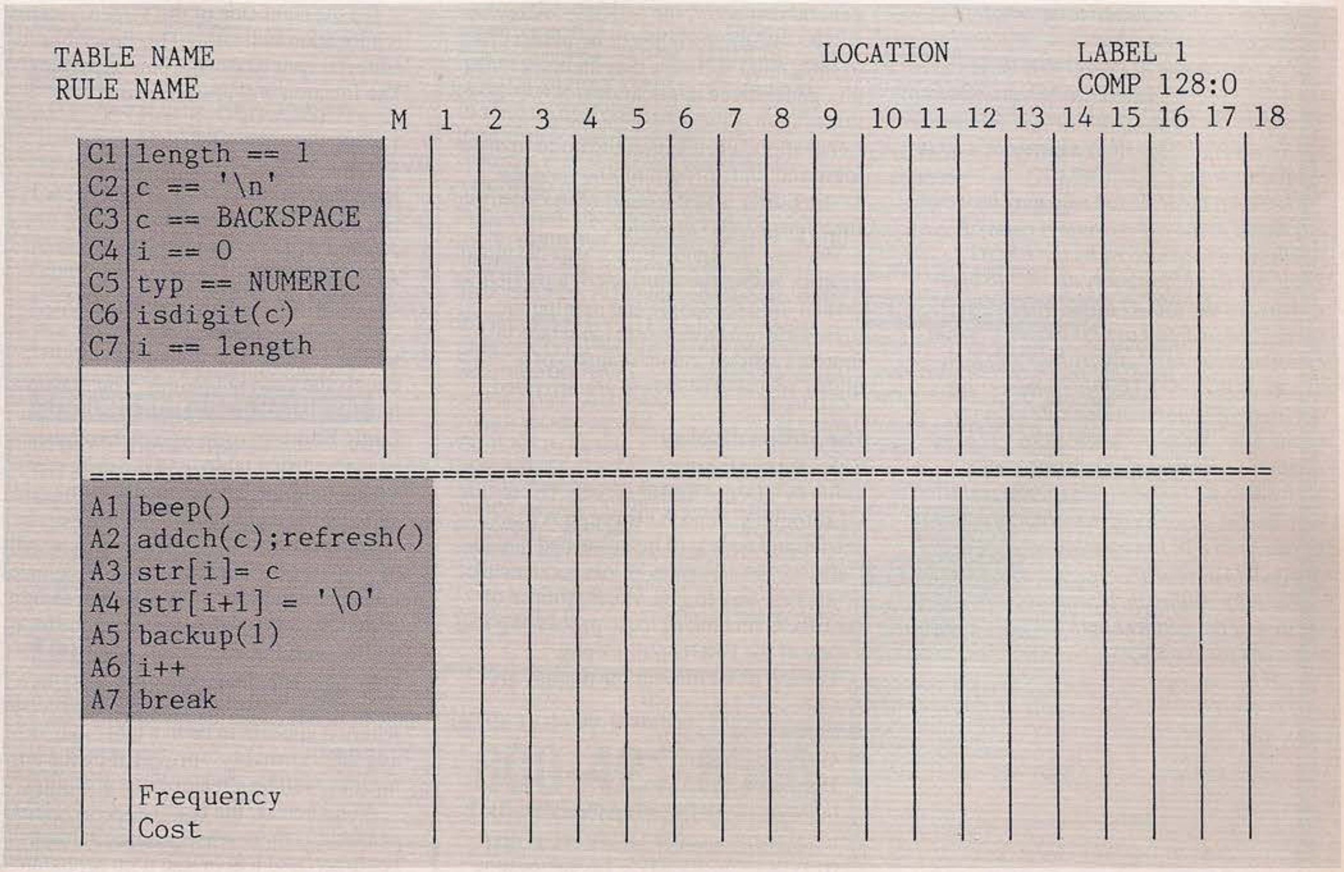
This is an important principle to grasp in understanding the aims of the ERGO Logic Kit. ERGO really just casts some arbitrary logical essence into different and varied forms. It can be handy to have your computer focus on the transformations while you tend to the information. This was most vividly shown by Lotus 1-2-3 with numbers and formulas transformed to the medium of business graphics. Using your computer to perform transformations will be a continuing theme with software in general and in this particular discussion of the ERGO Logic Kit.

A final use of the rule names is that they can become embedded comments in the code produced as output by the ERGO Logic Compiler, and they can form a part of the display presentation of the ERGO Logic Interpreter.

To the right of the rule name indicator is blank area that will display the matrix values, which will be discussed later.

To the right of this area is a key status indicator carrying the label "COMP:". This is the table completion indicator. It is expressed as the ratio of distinct simple rules that *should* be in the table (as computed by the matrix arithmetic) vs. the number that *are* in the table at any given instant.

A blank table will properly have 0:0 as the constituents of this ratio. By way of example, a limited entry (y/n) table with



```
TABLE NAME                               LOCATION        LABEL 1
RULE NAME                                                COMP 128:0
                       M  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
  C1|length == 1
  C2|c == '\n'
  C3|c == BACKSPACE
  C4|i == 0
  C5|typ == NUMERIC
  C6|isdigit(c)
  C7|i == length
  ==========================================================================
  A1|beep()
  A2|addch(c);refresh()
  A3|str[i]= c
  A4|str[i+1] = '\0'
  A5|backup(1)
  A6|i++
  A7|break

    |Frequency
    |Cost
```

Figure 1.

four conditions that is half completed will have as its completion ration 16:8 (16 = 2^8:8 = ½ of 16).

The user's continuous access to this little ratio is an important piece of the ERGO Logic Editor. This is because of the comfort humans generally gain when they know how far along they are in completing a task. The COMP ratio is instantly and always updated (at significant CPU cost, incidentally) as rules are entered and/or edited. The user's goal is to make those two numbers on either side of the colon match. With each little rule addition and with every clever rule specification that increases the righthand number or diminishes that on the left, the user is immediately rewarded with a certain visual indicator of his or her increasing success.

Much like a bookkeeper who must make both sides of the ledger balance or the householder reconciling the checking account, a good deal of task direction can be maintained by foreknowledge of the end points. A delectable bit of satisfaction can be had when all the numbers do at last line up, and you are utterly certain that you are done and you have a high degree of confidence that the work is correct. Achievement and victory, order out of chaos.

Viewed in this light, working with logic tables acquires many of the elements that make playing a game play instead of work. Even a modest increase in the amount of pleasure to be had in creating and analyzing logical systems could entice many more people into recording their experience in more directly useful, logical forms. Such a development would have unpredictable though possibly massive effects. So perhaps those dimensions of ERGO (or other comparably intended systems) that can be thought of as pleasurable are the most significant ones.

The first display segment on the third line from the top is labeled, simply, "M". This is the matrix column, holding the integer (2-9) that expresses the number of states a given condition can attain. This number defaults to two in the Logic Editor, as all logical problems of the class treated here can be reduced to a binary matrix, yes or no, 1 or 0. The Logic Editor encourages the use of extended entries (those with greater than a binary state) but does put an arbitrary cap of nine on the number of matrix values a single condition can attain.

The rest of the third line is given over to a rather mechanical enumeration of the rules. The alphanumerical sequence proceeds 1-99,A1-Z9,a1-z9,AA-zz. I do hope you never find yourself working on a table that requires (in its final form) more than 99 distinct rule expressions. You will have tackled a very formidable problem indeed.

Remember that a single complex rule can cover dozens, even hundreds, of simple rules. So while it is likely that you will encounter tables with greater than 99 simple rules, they will probably condense down to less than a dozen complex rule expressions.

The table image

The table image occupies the space starting with the fourth line of the display and continuing across and down the balance of the screen area.

The leftmost columns of the top half of the table are the label fields. By convention, one labels conditions C1-Cn, in numerical sequence. Sometimes other labels are appropriate, particularly directives to the compiler or the interpreter.

The next field is the stub, or description field. While the actual screen space allotted for each of the stubs is (by default) 12 characters in width, each stub field is actually a horizontally scrollable window into a line of 50 characters. You can set the number of characters that will be visible in this field.

When the ERGO Logic Kit is used as a programmer's tool, these stubs are presumed to contain syntactically meaningful target source language expressions. Of course, the user is free to type in whatever is desired, but it is possible to make the ERGO Logic Compiler output look pretty nonsensical and certainly uncompilable by a destination compiler.

Nonetheless, pseudocode type of expressions are often used here, as what is desired from ERGO is a module template, or architectural drawing as it were, rather than a compilable module.

The next array of fields on the top half of the table image is the matrix count for each condition stub. While the maximum matrix is only nine, I personally have yet to encounter a problem that both required more than nine matrix values and couldn't be more reasonably expressed in another way.

The condition entries themselves comprise the rest of the top half of the table. A description of legal table entries is in order.

Legality in a table entry is a function of the matrix of the condition to the left of that entry. If the matrix is only two, there are only three legal entries: y/n/— (the dash, of course, meaning "don't care"). If, on the other hand, the matrix is greater than two, say four, legal entries are:

a/b/c/d/A/B/C/D/—

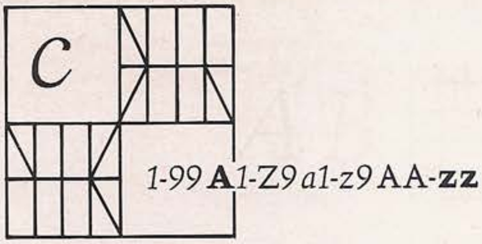
The lowercase letters read "is equal to (var)," the uppercase letters read "is NOT equal to (var)," and the dash continues to read "don't care."

The reason for including the "not equal to" expression is that it provides one more way for the user to keep the total number of rule expressions down. Consider: if you had a rule that held true in all cases but one, and there was no "not equal to" expression, you would have to explicitly key in (and document, and test, and maintain, ad infinitum) three affirmative rules (given a matrix of four) instead of a single negative rule.

It has been a nagging concern to me that I resorted to the terse a/b/c/etc. class of mnemonics for extended entries rather than some more comfortable natural language phraseology. For example, the phrases "low, medium, and high" make for eminent readability and clearly intensify the legibility of the table format.

Or perhaps the user would find more meaningful a construct like "always, often, sometimes, hardly ever, never." Of course, there is an infinite variety of descriptions that could be used that would be, in terms of sheer sensibility at least, superior to the single letter variable I have allowed.

The reason I chose a model with less



apparent communicative value was a pure and simple hunger for screen space. I decided, rightly or wrongly, that being able to see an entire table at once, without a lot of scrolling folderol, or waiting for a printer, or other less than optimal view port techniques, was ultimately more communicative, at a slightly higher level of abstraction, than the more full-blooded textual approach I might have implemented.

There is massive value in viewing a logical system in its entirety. The uniquely human capacity for indentifying structures can be brought into play: obscure patterns emerge, insights flash, and new levels of understanding open suddenly in the mind of the attentive user.

When you see that your logic looks like something, that there is an apparently natural design to what you suspected was a disordered and peculiar thought process, the balance shifts in your favor a bit. You begin to realize that this problem is soluble, it's merely a matter of arranging the patterns properly.

Though there may be moments of frustration ahead, your unconscious concerns about control and the doubtful outcome of your efforts abate, your self-possession is more profoundly realized, and the program becomes an instrument you can play. The intimations of pattern are the point of identification between the user and the logic. These can be revealed by the display of the Logic Editor, and the recognition of coherence can be most pleasant and informative.

As is often the case, manifest patterns are the fruit of abstraction, and an unfortunate by-product of abstraction is often a lack of clarity. To preserve the optimal degree of abstraction without inducing too much subtlety into the system, I decided to encourage the user to assign names to the a/b/c/etc. condition entry symbols.

My encouragement takes the form of reasonable default values for extended entries. These are available for display and alteration at the command level and pop up as appropriate on the status display to remind the user of the meaning of a particular condition entry variable.

Moving down the table image of Figure 1 and jumping across the horizontal bar

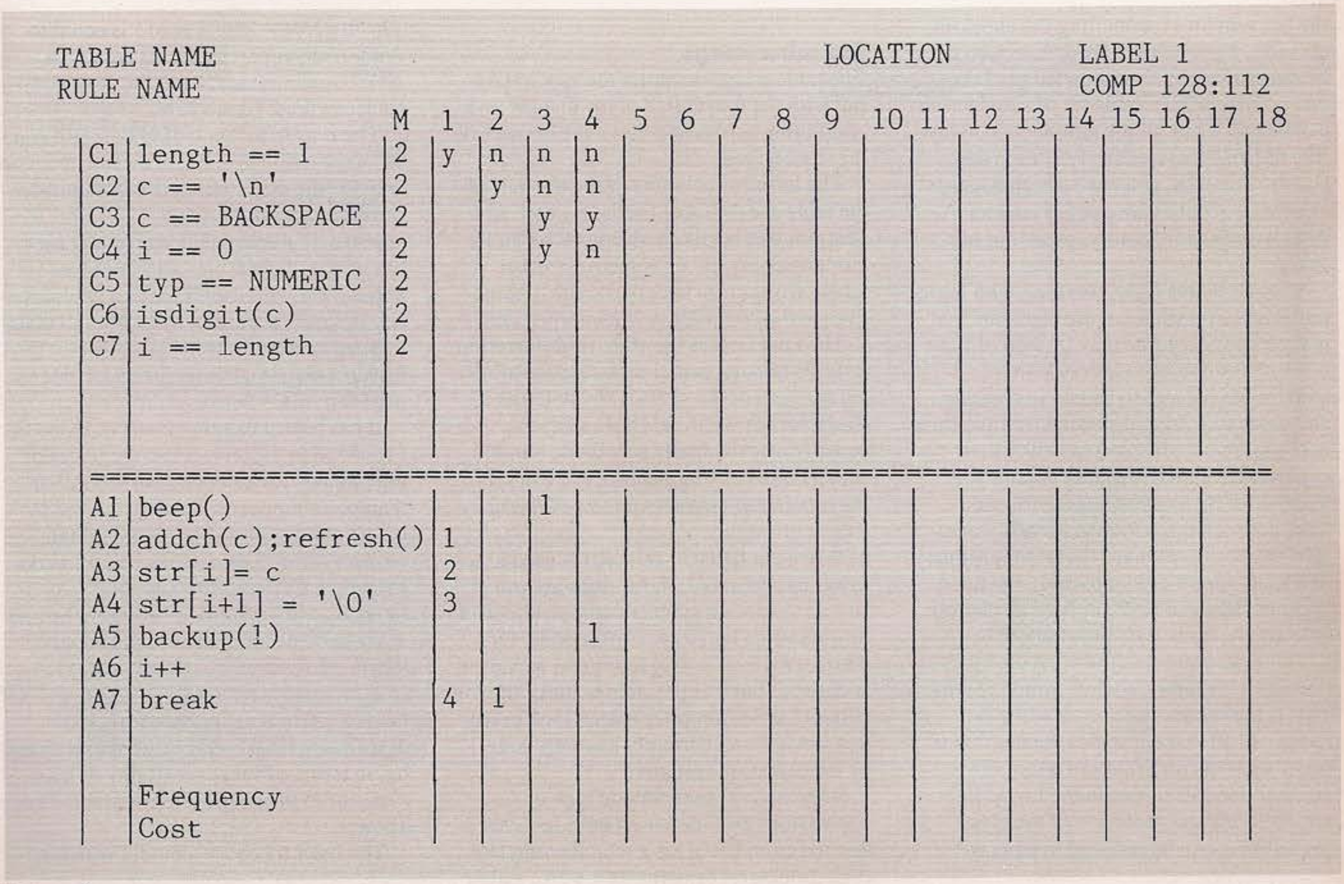

```
TABLE NAME                                             LOCATION        LABEL 1
RULE NAME                                                              COMP 128:112
                             M  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
  |C1|length == 1          |2 |y |n |n |n |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C2|c == '\n'            |2 |  |y |n |n |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C3|c == BACKSPACE       |2 |  |  |y |y |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C4|i == 0               |2 |  |  |y |n |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C5|typ == NUMERIC       |2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C6|isdigit(c)           |2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |C7|i == length          |2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |  |                     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  ===================================================================================
  |A1|beep()                  |  |  |1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A2|addch(c);refresh()      |1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A3|str[i]= c               |2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A4|str[i+1] = '\0'         |3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A5|backup(1)               |  |  |  |1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A6|i++                     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |A7|break                   |4 |1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |  |                        |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |  |Frequency               |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
  |  |Cost                    |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

Figure 2.

c d

aaaaaaaa

that distinguishes the condition from the action portions of the table, we see more or less the same layout: labels, stubs, no matrix for actions, and action entries.

Action labels are generally A1-An, again keeping in mind that certain ERGO Logic Interpreter directives, discussed later, can appear here. Action stubs are treated the same as condition stubs, and legal action entries are numeric in nature.

These numbers in the action entries (0-9) indicate precedence of action. That is, in a particular rule, the actions are executed in numerical rather than physical sequence (if there are more than 10 actions under a rule, you can number all the actions greater than nine with the number nine, and they will be executed in physical sequence).

Below the action segment of the table are two rows that allow the assignment of numerics to rules. These two lines are labeled "Frequency" and "Cost."

The frequency line enables you to enter a sequence of numbers that suggest how often a rule might be expected to be invoked. I have deliberately avoided calling this row "Probability" because that really gets us into another matter altogether.

Although the subject of probability and decision tables is a fruitful subject to pursue, and all types of benefits can be accrued by the judicious use of probabilities, time and space do not allow for an excursion into that area at present. Suffice it to say that the use of probabilities in the ERGO Logic Kit is awaiting a fuller treatment.

The optional entry of numbers into this row enables the ERGO Logic Compiler to do some optimizations relative to execution speed of the code structure produced (see section called The ERGO Logic Compiler) and can be used to specify to the ERGO Interpreter how the decision tree(s) should be constructed. Also the interpreter can make best guesses when relative frequency is known.

The cost row provides the opportunity to enter some numbers that indicate the degree to which the respective rules impose a burden, whether in code cost, or actual fiscal cost, or gravity, or any class of what we might loosely term "liability." These numbers likewise come into play at compile and interpret time.

The entries to both of these rows may (probably will) be somewhat loose. For example, one need not worry that the frequency entries add up to 100. The use of the numbers is to establish a kind of relativity among the rules. That is, "Rule n is more frequent (or costly) than Rule f, and Rule f is more frequent than Rule g, etc." Thus, it is important that we know only a rough approximation of frequency or cost, which, after all, is the only type you are likely to have available in most instances.

The command menus

The slash (/) key invokes the ERGO command menu tree. At the highest level, the menu offers the following options, displacing the top two status display lines:

File Table Draw Grid Compile Interp
Prefer Exit Quit

These options represent the major components of the ERGO Logic Kit. Selecting these options is accomplished by either striking the first letter of the option name or spacing through them with arrow keys or the space bar and hitting return when the proper command is highlighted. On the second line of the menu is either the appropriate submenu that will be invoked with the highlighted selection, or explanatory information relative to the highlighted command.

This menu-driven interface is obviously deeply indebted to Lotus 1-2-3, which, in its turn, owes much to Visicalc. While I make no claims that it is optimal, I really find myself without motivation to tamper with an interface that has proven more or less acceptable to millions of users over a relatively long period of time. Perhaps the single biggest barrier to overcome in deploying new software is training: users, dealers, distributors, etc.

By more or less cloning the interface that most people seem familiar with, perhaps much inertia can be overcome, and much more logic will be fruitfully encoded. Additionally, there was (and is) simply too much raw functionality to achieve, too many functions to invent, and too many things to do that haven't been satisfactorily done to devote energy to the creation of something that has.

Since we are now focusing on the editorial functions of the ERGO Logic Kit, the menu tree we will refer to is that which appears with the selection of the "Table" option from the main menu.

The getl table

We will do a small but probably useful example of a single table module to demonstrate how a programmer might make use of the ERGO Logic Editor. The specifications of our problem are such as are often encountered by programmers writing an interactive application.

We need a handy little input routine for miscellaneous *get*s from the console. We will pass this routine a string pointer, a type designator (*NUMERIC* or other), and a length indicator. *getl* (the name of our routine) must provide the following facilities:

- If the type is *NUMERIC,* accept only numbers from the keyboard; if anything else is entered (except as described later), ring the console's bell to indicate an error
- If the length expected is only one character, don't require that the carriage return be struck to indicate end of entry—rather, return to the calling program after the entry of a single character
- If the user has typed in *length* characters, don't accept any more (except a new line), instead, beep the console to indicate an error
- Accept a new line at any point to indicate end of entry
- Accept a backspace at any time, other than when the user hasn't entered anything yet, and call an appropriate backup routine
- Echo all legal output to the screen.

Since our ERGO Logic Compiler has an affinity to the C language, we will build the *getl* table from C language

CCCD

```
Condition specifications:

length of only one
'\n'
BACKSPACE
        except when i = 0;  /* no input */
NUMERIC typ
        require numbers for input
length exhausted

Condition code:

length == 1
c == '\n'
c == BACKSPACE
i == 0
typ == NUMERIC
isdigit(c)
i == length

Action specification:

ring the bell on errors
echo the characters typed
store the characters typed in the string provided
   implied is proper termination of the string
   and keeping track of where the current char is
backup on legal BACKSPACEs
return when finished

Action code:

beep()
addch(c);refresh()   /* 'curses' screen update */
str[i] = c
str[i+1]='\0'
backup(1)     /* function that manages a backup */
i++
break         /* we'll assume an infinite loop */
```

Listing 1.

expressions. We will assume that the variables passed to us are:

```
char str[];  /* destination string for */
             /* keyboard input */
int typ;     /* NUMERIC or other */
             /* input type indicator */
int length;  /* maximum number of */
             /*allowable characters */
```

The calling program will be held responsible if there is insufficient room in the string pointer or if the type or length indicator are radically wrong or out of bounds (such as a length greater than available memory or a zero or less then zero length). We will also assume two local variables:

```
int i = 0;     /* index pointer */
char c;        /* the current character */
```

to help us keep track of how far along in the string collection we are and what it is we have on hand in the current character.

The specification tells us to watch for the code (and its ramifications) as presented in Listing 1.

Entering the conditions and actions as described into the ERGO Logic Editor, we have the working table shown in Figure 1. Note the large number of simple rules yet to be created in order to attain complete mechanical perfection: 128. Let's see what we can do to cut that daunting number down to size.

We'll start with some of the easy rules that come to mind at once. For example, if $i == 1$, we always store the *char*, terminate the string, and return. Another easy rule is if $c ==$BACKSPACE and $i == 0$ —there's nothing from which to back up, so we beep the bell. Alternatively, we do back up whenever there's a BACKSPACE and i does not equal 0). Also, if c $==$ '\n', then we're all done, no matter what.

We enter into the Logic Editor both condition and action entries that express this logic, as in Figure 2. Note that with these four complex rules (spaces are interpreted by the Logic Editor as dashes if there is an entry on the top condition row) we have completed nearly 90% of the table. Also note the n entries that were only implicit in our textual description of the rules (three paragraphs previous) at,

A

for example, entry 1:2 (condition 1, rule 2). This is commonly done to distinguish rules and to make manifest implicit exclusion.

Another interesting point here is the stair-step image of the logic. This also is common and represents (at least to my mind) a well-ordered analysis of the hierarchy of the logic. Such stair-step images are often produced by the sorting operations available in the Logic Editor.

Now let us assume that all the easy rules are accounted for. There remain 16 rules yet to be entered. This can be quite tedious work, and here is where the Logic Editor really lends us a hand by generating the missing rules for us either one at a time or all at once.

Selecting the *Missing* command from the table submenu, we can then have ERGO fill out all the missing rules for us. We add dashes as appropriate and use the ERGO disambiguate function to eliminate redundant columns.

The case of disambiguation presents an interesting problem: if two rules overlap, how is the Logic Editor to know which of the two is the superfluous rule? After all, it *must* pick one of them for elimination.

For this, the ERGO Logic Editor employs the following algorithm:

- Compare the column counts of the rules (which will be either the product of the matrices of the dashed entries, or if there are no dashes, 1)
- If one is less than the other, delete the rule with the lesser column count
- Otherwise, delete the rightmost rule.

Whenever a programmer automates behavior that normally requires judgement, extreme caution is indicated. The judgement automated here is that a rule with more information (that is, column count), is somehow better than its less informative neighbor and more deserving of continued existence.

Additional judgement is rendered that the ERGO user is working from left to right and the sequence of table creation we just used in our example will emerge as the standard way to create tables. Since the user will be entering his or her rules close to the left-hand boundary of the table and since ERGO adds its rules to the end, or the right side of the table, the rightmost should be the one to go.

The rationale for this particular set of judgements is that simple (nondashed) rules are cheap. The Logic Editor can generate them effortlessly, one at a time or in large groups. But a complex rule, with dashes or NOT entries, is a rule that has the human element. It has necessarily been born of some person's mind and is therefore more valuable (at least from the human viewpoint).

The real problem emerges when two human-generated rules experience friction. As a result of this tidy little algorithm, one of them will surely die.

Picture the following case: there are three rules, two complex and one simple:

1	2	3
y	y	y
n	n	—
—	y	n

Upon automatic disambiguation, the following events take place:

- R1 has a greater column count than R2, and therefore R2 is removed
- R1 and R3 match column counts, but R1 is the leftmost rule, so R3 is extinguished.

All this is well and good, I suppose, but the problem is that *I liked R2 and R3*: between them, they had a higher column count than R1, and I figured the Logic Editor would remove R1!

While *Undo* is a function that expresses the lack of intelligence in our software, I am afraid that it was this rule-gobbling disambiguate function (along with the system's blithe ignorance of operations the user orders up but *can't* really want) that caused me to create an *Undo* operation for the ERGO Logic Editor.

This is a brute-force *Undo*. With each radical table operation (*Delete/Insert/Sort/Disambiguate*/etc.), a total copy of the current table is made. This only goes to one level, and it takes time and space, but it does have the virture of restoring a table that has been botched.

An *Undo* function can also cover a multitude of sins. Excessive prompting and verifying, a practice not unlike the doctors' practice of defensive medicine, can be minimized; user temerity can be reduced; and the revocability of major operations encourages their use.

Not one to overestimate either ERGO's perfection or its ability to make clear what might happen next, I also extended the *Undo* command to *Undo* itself. That is, if I like what I have undone less than what I had done, I can undo what I have undone and do what I had done.

Next month we will complete the logic processing associated with the *getl* table, compiling and optimizing it in various ways. Also we will delve into the Logic Interpreter and develop a small expert system.

References

1. Pollack, S.L. "Conversion of Limited-Entry Decision Tables to Computer Programs." *CACM* 8,11 (Nov. 1965): 677-682.

Jim McCarthy is a member of the technical staff, research and development, AT&T-Teletype, in Skokie, Ill.

Try out the ERGO Logic Kit

You are invited to experiment with the ERGO Logic Kit.

I have two motives for distributing these parts of the ERGO Logic Kit. I hope that the software will be useful, and I am looking to "prove-in" both the software itself and the ideas that engendered it. You can communicate with me by sending letters to: Jim McCarthy, c/o *COMPUTER LANGUAGE*, 131 Townsend St., San Francisco, Calif. 94107, or via UNIX mail at : ihnp4!ttrdc!jm. I will attempt to respond to all communications I receive.

You should be aware that I am not placing the ERGO Logic Kit, or any part of it, in the public domain. With every diskette order that the *COMPUTER LANGUAGE* Users Group receives, I will receive $.10 as a license fee. You in turn will receive a license that allows you to do essentially whatever you desire with the software except sell it or otherwise distribute it. You are encouraged to return the completed license so that good records may be kept from the start.

You should also be aware that the ERGO Logic Kit is more a prototype than a product, with all that implies.

Please give the kit a try, and let me know what you think and what bugs you find.

Write it once!

MasterFORTH

Portable programming environment

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest. If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

IBM TM

MasterFORTH is a state-of-the-art implementation of the Forth computer language. Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in macro assembler to make it even faster. MasterFORTH's relocatable utilities, transient definitions, and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint, and trace your way through most programming problems. A string package, file interface, and full screen editor are all standard features.

CP/M TM

MasterFORTH exactly matches the Forth-83 Standard dialect described in *Mastering Forth* by Anderson and Tracy (Brady, 1984). The standard package includes the book and over 100 pages of supplementary documentation.

MasterFORTH standard package	
Macintosh	$125
IBM PC and PC Jr. (MS DOS 2.1)	125
Apple II, II+, IIe, IIc (DOS 3.3)	125
CP/M 2. x (IBM 3740 8")	125
Commodore 64	100
Extensions	
Floating Point (1984 FVG standard)	$60
Graphics (Apple II series)	60
Module relocator (with utility sources)	60
Printed source listing (each)	35
Publications	
Mastering Forth (additional copies)	$18
Thinking Forth by Leo Brodie	16
Forth-83 International Standard	15
Rochester Bibliography, 2nd ed.	15
1984 Rochester Conference	25
1984 FORML Conference	25

MICROMOTION

12077 Wilshire Blvd., #506
Los Angeles, CA 90025
(213) 821-4340

CIRCLE 64 ON READER SERVICE CARD

TOTAL CONTROL with *LMI FORTH*™

Z-80 FORTH
8080 FORTH
68000 FORTH
8086 FORTH
PC/FORTH
PC/FORTH
LMI Metacompiler

For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, and 6502
- No license fee or royalty for compiled applications

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI ***Laboratory Microsystems Incorporated***
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, D-7820 Titisee-Neustadt
UK: System Science Ltd., London EC1A 9JX
France: Micro-Sigma S.A.R.L., 75008 Paris
Japan: Southern Pacific Ltd., Yokohama 220
Australia: Wave-onic Associates, 6107 Wilson, W.A.

CIRCLE 48 ON READER SERVICE CARD

PUBLIC DOMAIN SOFTWARE REVIEW

dBASE II programs

By Tim Parker

Three application programs dominate the microcomputer field, both in home and business use: word processing, spreadsheet (financial) applications, and data bases. While discussion of products for the first two categories is wide open (witness the large number of word processing packages and spreadsheets that vie for attention), the data base end has been dominated by Ashton-Tate's dBASE line.

Though there are many other data bases, some of which may be better than the dBASE line, if ever there was a typical package for any of the three application programs, dBASE fits the bill for data bases. But whether dBASE is the best data base available is immaterial for our purposes here; we're more concerned with the growth of dBASE public domain support.

Public domain dBASE programs tend to come in two varieties: utilities and standard applications, such as mailing lists. Although the dBASE public domain programs don't really match those of a custom programmer's efforts for a particular need, most of the material is of reasonable quality. In many cases, these programs are quite adaptable for several uses.

Discussing dBASE requires considering a number of categories, including a comparison of dBASE II and dBASE III. dBASE II began as a standard program in the 8-bit CP/M world and was rapidly adapted to 16-bit machines. dBASE III soon followed, taking greater advantage of 16-bit capabilities. In fact, dBASE III is probably one of the most copied programs available, with the possible exception of WordStar and VisiCalc.

But while dBASE III is relatively new, there is far more available for dBASE II, so we'll concentrate on the latter. There is also the matter of target machine: 8- or 16-bit. Both programs will run on either because the code is transportable, but the issue here is disk availability and formats. Unless you have one of the excellent series of multiformat programs such as UNIFORM, your machine is probably limited to one or two formats.

To begin our discussion of dBASE II, let's delve into the world of Big Blue and the IBM format public domain dBASE II. Most of the programs to be considered in this column surfaced originally as CP/M versions but now have been transferred to the MS-DOS format.

When tackling IBM disks, the first reference I inevitably check is the PC-SIG catalog. Several IBM disks have dBASE material available, some dedicated, some mixed in with other programs garnered from bulletin boards, CompuServe, and the like. As mentioned earlier, these programs tend to fall into two categories: completed applications and utilities and "boilerplate" routines that can be rapidly patched into code for a particular purpose. The latter are probably more useful for programmers.

An excellent example of the so-called boilerplate routines appears on PC-SIG vol. 109. One routine checks whether dates entered into a program are valid or not. This routine ranges from simple (checking for invalid characters) to more complicated (checking for November 31) to the most complex (whether February 28, 1984, occurred).

One fairly common approach for checking dates is to convert to Julian days, use algorithms to check correctness, and then reconvert. Alternatively, recent years can be checked by comparing matched substrings of the date function with a list of correct values. This is the approach taken in DATE.PRG (or .CMD on 8-bit), written by Douglas Haman. This routine is self-contained and the results easily imported and exported. The date entered or passed to the routine is broken down into month, day, and year substrings and the results matched against a list of valid characters.

The routine is not particularly short, involving several nested *DO* loops and a mess of *CASE* statements, but it does the job. Documentation is nonexistent, but it is unnecessary for all but inexperienced dBASE users. The code is structured just as the manuals suggest and requires only slight modification to fit easily into a program, but some programmers might want to rewrite sections to match their programming style. As a quick and dirty routine, it saves time in thought and coding.

Another problem that often occurs when displaying data base fields on the screen is a wraparound of the field when it is lengthy. Most programmers get around this by limiting the fields requiring this sort of space (such as a comments field) to 60 characters or so and having multiple fields devoted to the space (COMMENT1, COMMENT2, etc.) on separate lines. However, there are times, such as when a lot of fields are in use, when it is necessary to have a long field. When displayed on the screen or printer, a long field will break at an arbitrary place depending on the display space available.

dBASE does not have word-wrap or anything similar. Usually it is necessary to try to arrange the displays so that a break does not occur in the middle of a word. A program by Melissa Gray on PC-SIG vol. 139 allows a long field to be broken at a blank character or some other allowable line break character, such as a dash, semicolon, comma, equal sign, or backslash, and wrapped onto the next line. This program, called PRINTXT1, is one of three files available on the disk. The other two are a documentation file and a procedure file.

Records to be printed are trimmed so that extra space is deleted and blank records not displayed. A series of variables require initialization before the routine is used, depending on the context.

The coding itself is not particularly elegant and can probably be compressed quite a bit if desired. The author admits that it is a beginning effort. Regardless, it does the job it is intended to do and solves the common dBASE problem of wrapping.

PC-SIG vol. 143 has a program that should be of interest to dBASE users. COPYFLD, written by Kenneth Eagle, provides a more powerful *JOIN* command function and allows a third file to be created by appending transferred fields rather than using dBASE's *JOIN* command. Further, two or more fields in the source data base can be compressed into one field in the target data base. A documentation file is provided, but the program is documented well enough to be figured out without it.

The program displays prompts for the name of the source file and destination file. (In a completed application, this process can be bypassed with macros.) Prompts for the fields to be copied to and from are then displayed. The program is somewhat lengthy but checks for errors. Indexed files are used throughout. Depending on the case, both the target field and the field that is to be copied from must be the key index.

I used the COPYFLD program quite a bit for testing when creating a large inventory data base program involving 12 data bases. (Most were combinations of other data bases but for different purposes and broken up for speed—a necessary constraint in this particular project.) No problems were encountered, and some tedious and frustrating work was eliminated. This program would be very useful for large data base programmers working in dBASE II.

Vol. 128 in the PC-SIG library includes several dBASE utilities. Programs that determine the day of the week and set the system time and the date variable from the system date are useful routines that can be included in other coding. COMSTRIP is a quickie routine that strips comment files from other dBASE II programs. The advisability of such a move is left to the user, but where size is a factor such a move is probably necessary.

A series of completed applications are available on PC-SIG vol. 126. The disk contains seven different program series: a form letter generator, a mailing list manager, two backup routines, a library routine, a state and zip code checker, and a conversion program, with documentation.

Warren McKenna's BACKUP is CP/M-based and relies on the *QUIT TO* command to use PIP (the peripheral interchange program). It can be adapted to MS-DOS by using the *COPY* command instead, but the structure of MS-DOS when compared to CP/M may not allow the program to work as easily. It adapts to the CP/M version through a relatively short routine that utilizes file transfers, but it cannot handle files larger than disk capacity.

For large files, SUPERBAK (by an unnamed contributor) uses hard disk backup onto floppies. Though the program is longer it works well, allowing the user to name the input and output files or data bases. This routine works in CP/M or MS-DOS, although you have to insert a *RESET* statement in CP/M versions whenever disks are changed.

SUPERBAK prompts the user for the capacity of the floppies to be used and requires the number of characters per record to be inputted. (This can be read directly from the records if the coding is changed slightly.) Restoration from the backups is also easy. This routine can be readily incorporated into another program by using the *BACKUP/RESTORE* option on a menu and looping through a *DO* to this program.

STATEZIP is a program (author unidentified) that checks for valid state abbreviations and correct zip codes. It includes a data base with (surprise!) 52 entries listing state abbreviations and two fields three digits wide with the high and low valid zip codes.

The program is a self-contained procedure and can be trimmed to fit most programs. Variables may have to be renamed for usage optimization—especially for variable passing—but the program is quite well laid out and concise enough to be useful in any program incorporating zip codes.

In use, the program asks for the data base to be checked. This process can be avoided if the program is included in a larger routine. The data base to be checked is called the SECONDARY file, while the STATEZIP used as a reference is the PRIMARY file. After the program has been executed, you can edit the erroneous data or print out the bad values for future reference.

MAIL is a series of programs that allows a mailing list to be generated and printed on labels and form letters to be used in conjunction with the mailing list. Mailing lists, as many custom programmers know, are one of the most frequently requested and easiest programs to devise. Keeping a few basic versions on the shelf

Realia COBOL

What to do while your COBOL programs compile and execute:

1. Wait.
2. Wait some more.
3. Stop waiting. Call Realia.

Patience isn't always a virtue.
Realia COBOL is **fast:**

Compilation Speed (minutes:seconds)

Lines in Program	Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1,000	:51	8:33	3:42	5:05	5:11
5,000	3:30	48:07	16:58	*	45:26

*Could not successfully compile the program.

Execution Time Ratio
(Gibson Mix; calculated S-Profile)

Realia COBOL	mbp COBOL	Level II COBOL	R-M COBOL	Microsoft COBOL
1.0	3.6	14.7	21.6	22.3

Sieve of Eratosthenes
0.818 seconds per iteration

All benchmark tests were performed on an IBM PC-XT with 192KB of memory. IBM PC-XT is a registered trademark of International Business Machines Corporation, mbp COBOL, of mbp Software and System Technology, Level II COBOL, of Micro Focus, R-M COBOL, of Ryan-McFarland, and Microsoft COBOL, of Microsoft.

Realia COBOL is written in COBOL. We offer you the tools we use ourselves:

- Our FOLLOW-THE-SOURCE™ interactive symbolic debugger. Works with normal native code.
- A speedy full-screen editor that handles very large files.
- Mainframe IBM VS COBOL compatibility.
- Interfaces to Assembler and C.
- No royalty or run-time fee.
- No limit on program size, up to available memory.
- In our new release, no need to insert the product diskette when you're using a hard disk.

Realia COBOL costs $995. Qualified companies can try it for free. Call us. And ask about our other products, Spacemaker™ and Termulator™.

What are you waiting for?

REALIA inc.
10 South Riverside Plaza
Chicago, Illinois 60606
(312) 346-0642
TELEX: 332979 (REALIA INC)

CIRCLE 76 ON READER SERVICE CARD

is something a dBASE programmer shouldn't pass up, as these can quickly be tailored to a customer's requirements. MAIL does a fairly good job of creating lists but lacks any editing functions for list maintenance. These have to be added by the user.

MAIL's design follows the form letter approach and uses a standard letter format that can be edited with a word processor or dBASE itself. The letter salutation and headers are contained in a file with the dBASE commands for merging with the rest of the data base, while the body of the letter is in a separate file, saved as question mark statements.

The mailing list itself is straightforward, with fields for a title or salutation, surname and given name, address, company and title, and state and zip code. The entry process is simple but has no real power for editing, manipulation, or checking entries.

The label generator is routine, and the printout uses a predefined format. It would be nice to have variable spacing, width control, and fields printed, but these would make the program more complex. Also, selective printouts of matches with defined fields (such as all California residents or people of ABC company) would be appreciated by most mailing list users.

The routines work as they should, but the program as a whole is rather rudimentary, suitable for small uses but definitely requiring programmer modification for any large-scale usage. The series as a whole seemed remarkably like the mailing list and form letter routines included in the sample programs with dBASE II, particularly in terms of capability.

A more sophisticated mailing list and form letter generator are included on the same disk under the name ML2. This program, written by Wil Wakely for the San Diego Computer Society, is far superior in all respects to MAIL, except for the user whose major concern is simplicity.

ML2 allows the user to edit entries. The data base is composed of fields for name and address, home and work phone numbers, and three fields—SORT, EXPIRE, and TYPE—for membership lists or categorizing data base entries.

Form letters use the same format as the previously mentioned programs, with a word processor used to create the body and the letter read into the print routine. The label printing routine allows multiple labels to be printed only in the standard (3 ½-in.) label size.

A disk library index manager was contributed to this volume by Dr. M. Timin. This application allows the creation of a subject index to a disk file library, with brief descriptions of passages referred to in a separate data base. The user can search throughout the data base contents for a string match or a particular subject. For those with extensive disk libraries, this may be of some interest.

PC-SIG vol. 102 has a complete series of programs designed for advertising agencies. These are the sample programs included in the dBASE II manual, but revised, modified, enhanced, and entered on disk for general use. While there may be limited demand for this particular type of program, it can be adapted easily to other situations and shows the construction of a medium-sized application.

Provisions for entering bills (and paying them), time sheets, deposits, check maintenance, media insertion orders, client billing, and various status reports are included. The program also has job-costing potential.

In addition to the products listed here, much more dBASE material is available on the PC. Many of the programs mentioned in this column, as well as some not yet listed by PC-SIG, also are available from the New York Amateur Computer Club's PC/BLUE Library. Many of these programs are available in CP/M format from several sources, including SIG/M, CPMUG, and public domain libraries around the country.

Next month we will be publishing a list of sources for public domain material, including addresses, prices, and (where possible) phone numbers. One such source is available from Dynacomp. This company has released a catalog that includes several public domain library sets in addition to its line of commercial software. This catalog (#28) is available for $1.

Apple software is represented on 50 disks culled from the over 100 available in other collections, such as the Public Domain Exchange. Priced as low as $2.95 in quantities of 10 or more, this is probably the cheapest public domain library I know of for the Apple.

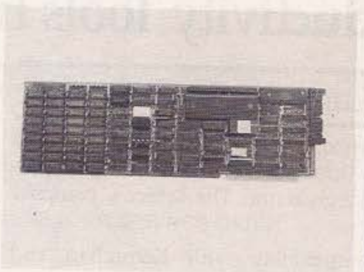
PC/BLUE and PC-SIG libraries are also available for purchase or rent. Thirty-two disks, representing their better volumes, have been culled from PC/BLUE. Individual disks are $5.95 each, while the full disk collection is $119.95. (Rental of a 32-disk collection for 10 days is $39.95.)

The PC-SIG collection is made up of 98 disks and can be purchased in its entirety for $399.95 or rented for 10 days for $99.95. Individual disks are $5.95.

CP/M is represented by a nine-volume collection available for $39.95 (10-day rental) or $153.95 (purchased) with a book of documentation published by People Talk. (The book is available separately for $17.95.) Individual disks are $4.95.

For further information, contact Dynacomp at 1064 Gravel Road, Webster, N.Y., 14580. ■

68000 CO-PROCESSING For IBM PC, PC/XT and COMPATIBLE SYSTEMS



Now you can add the MOTOROLA 68000 16/32 Bit Processor to your PC via use of the Pro 68 Advanced Technology Co-Processor. Enjoy all of the performance benefits of the 68000 processor without sacrificing your current PC system. Consider these impressive standard features of Pro 68:

- High Speed MOTOROLA 68000 micro processor
- 10Mhz no wait state design (3 times faster than the IBM PC/AT)
- True 16/32 bit technology
- For use on IBM PC, PC/XT or compatible systems
- On board 16 bit parity checked memory, 256K to 1024K
- Two serial I/O ports for multi user interface
- Provisions for the high speed NS32081 math processor
- High speed proprietary dual port host bus interface
- Parallel or array processing via multi processor architecture
- MS/PC DOS RAM disk driver program
- Choice of two popular integrated 16/32 bit operating systems:
  - CPM68K from Digital Research Inc.
    - Full suite of development tools
    - "C" compiler with floats and UNIX I/O library
    - Many third party compatible languages and applications
  - OS9/6800 from MICROWARE Corporation
    - UNIX look alike with multi user/multi tasking, shell, hierarchical disk directory, record and file lock, pipes and filters
    - Full suite of development tools
    - UNIX V compatible "C" compiler
    - Optional languages include BASIC, ISO PASCAL, FORTRAN 77.

Pricing from $1195 includes Pro68 with 256K, OS, and MS/PC DOS RAM disk driver. HSC also manufactures and markets a full line of co-processors and RAM disks for use on Z80 based systems.

DISTRIBUTORS:

Australia-Computer Transition Systems ...03-537-2768
Great Britain-System Science ...01-248-;062
West Germany-DSC International ...089-723-1125
Canada Remote Systems ...416-239-2835

Dealer, Distributor and OEM inquiries invited.

HSC

Hallock Systems Co., Inc.
267 North Main Street
Herkimer, NY 13350
(315) 866-7125

CIRCLE 12 ON READER SERVICE CARD

C is the language.
Lifeboat™ is the source.

Productivity Tools from the Leading Publisher of C Programs.™

The Lattice® C Compiler

The cornerstone of a program is its compiler; it can make the difference between a good program and a great one. The Lattice C compiler features:

- Full compatibility with Kernighan and Ritchie's standards
- Four memory model options for control and versatility
- Automatic sensing and use of the 8087 math chip
- Choose from the widest selection of add-on options
- Renowned for speed and code quality
- Superior quality documentation

"Lattice C produces remarkable code...the documentation sets such a high standard that others don't even come close...in the top category for its quick compilation and execution time and consistent reliability."

Byte Magazine

Lattice Library source code also available.

Language Utilities

Pfix 86/Pfix 86 Plus — dynamic and symbolic debuggers respectively, these provide multiple-window debugging with breakpointing capability.
Plink 86 — a two-pass overlay linkage editor that helps solve memory problems.
Text Management Utilities — includes GREP (searches files for patterns), DIFF (differential text file comparator), and more.
LMK (UNIX "make") — automates the construction of large multi-module products.
Curses — lets you write programs with full screen output transportable among all UNIX, XENIX and PC-DOS systems without changing your source code.
BASTOC — translates MBASIC or CBASIC source code directly to Lattice C source code.
C Cross Reference Generator — examines your C source modules and produces a listing of each symbol and where it is referenced.

Editors

Pmate — a customizable full screen text editor featuring its own powerful macro command language.
ES/P for C — C program entry with automatic syntax checking and formatting.
VEDIT — an easy-to-use word processor for use with V-PRINT.
V-PRINT — a print formatting companion for VEDIT.
CVUE — a full-screen editor that offers an easy way to use command structure.
EMACS — a full screen multi window text editor.
Fast/C — speeds up the cycle of edit-compile-debug-edit-recompile.

Graphics and Screen Design

HALO — one of the industry's standard graphics development packages. Over 150 graphics commands including line, arc, box, circle and ellipse primitives. The **10 Fontpack** is also available.
Panel — a screen formatter and data entry aid.
Lattice Window — a library of subroutines allowing design of windows.

Functions

C-Food Smorgasbord — a tasty selection of utility functions for Lattice C programmers; includes a binary coded decimal arithmetic package, level 0 I/O functions, a Terminal Independence Package, and more.
Float-87 — supports the 8087 math chip to boost the speed of floating-point calculations.
The Greenleaf Functions — a comprehensive library of over 200 routines.
The Greenleaf Comm Library — an easy-to-use asynchronous communications library.
C Power Packs — sets of functions useful for a *wide variety of applications.*
BASIC C — This library is a simple bridge from IBM BASIC to C.

Database Record Managers

Phact — a database record manager library of C language functions, used in the creation and manipulation of large and small databases.
Btrieve — a sophisticated file management system designed for developing applications under PC-DOS. Data can be instantly retrieved by key value.
FABS — a Fast Access Btree Structure function library designed for rapid, keyed access to data files using multipath structures.
Autosort — a fast sort/merge utility.
Lattice dB-C ISAM — a library of C functions that enables you to create and access dBase format database files.

Cross-Compilers

For programmers active in both micro and mini environments we provide advanced cross-compilers which product Intel 8086 object modules. All were developed to be as functional — and reliable — as the native compilers. They are available for the following systems:

VAX/VMS, VAX/UNIX, 68K/UNIX-S,
68K/UNIX-L

Also, we have available:
Z80 Cross-Compiler for MS- and PC-DOS — produces Z80 object modules in the Microsoft relocatable format.

New Products

Run/C — finally, a C interpreter for all levels of C Programmers.
C Sprite — a symbolic debugger with breakpoint capability.

*Call **LIFEBOAT:** 1-800-847-7078. In NY, 1-212-860-0300.*

C Programming Tools
Lifeboat
ASSOCIATES

YES! Please rush me the latest FREE Lifeboat™ catalog of C products.
*Name*__________ *Title*__________
*Company Name*__________ *Business Phone*__________
*Address*__________
Please check one of the following categories:
☐ *Dealer/Distributor* ☐ *End User* ☐ *Other*__________

Return Coupon to: Lifeboat™ Associates
1651 Third Avenue, New York, NY 10128

© 1985 Lifeboat Associates

CL

CIRCLE 87 ON READER SERVICE CARD

⊗○⊗○ PRODUCT BINGO

By Doug Millison

Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE readers. Send new product information to Doug Millison, Product Bingo, COMPUTER LANGUAGE, 131 Townsend St., San Francisco, Calif. 94107.

Ladder up to TopView

C program developers writing for **IBM's TopView** can use a new library of 70 C functions. For the IBM PC family, 512K memory suggested, the Lattice TopView Toolbasket retails for $250.

Lattice Inc., P.O. Box 3072, Glen Ellyn, Ill. 60138, (312) 858-7950.

CIRCLE 101 ON READER SERVICE CARD

APL a day, chapter 2

Users of APL*PLUS can speed program development under IBM DOS with **APL*PLUS PC TOOLS**, vol. 2; the first module is now available for $85.

STCS Inc., 2115 E. Jefferson St., Rockville, Md. 20852, (800) 592-0050 (in Maryland, (301) 984-5129).

CIRCLE 103 ON READER SERVICE CARD

AT&T PC UNIX compilers

AT&T users will develop business applications under UNIX with the **Philon** family of **FAST/** compilers, including **FAST/C, FAST/BASIC-C, FAST/BASIC-M, FAST/COBOL FAST/FORTRAN, FAST/PASCAL,** and **FAST/RPG.** Prices were unavailable at press time.

Philon, 641 Ave. of the Americas, New York, N.Y. 10011, (212) 807-0303.

CIRCLE 104 ON READER SERVICE CARD

New, low-priced Pascal

An interactive, multitasking Pascal compiler that claims to be a strong competitor to Turbo Pascal in speed, reliability, and price. **Mystic Pascal** is priced at $39.95, plus shipping.

Mystic Canyon Software, P.O. Box 1010, Pecos, N.M. 87552, (505) 988-4214.

CIRCLE 105 ON READER SERVICE CARD

16/32-bit coprocessor

IBM PC users work in both worlds with the **Pro68** coprocessor board based on the Motorola 68000 chip, with CPM 68K or OS9/68000 operating systems. Priced from $1,195 for the 256K version to $1,895 for the 512K coprocessor board.

Hallock Systems Co. Inc., 267 N. Main St., Herkimer, N.Y. 13350, (315) 866-7125.

CIRCLE 106 ON READER SERVICE CARD

BASIC to Pascal translation

Programmers can convert Applesoft BASIC programs to Apple Pascal with **P-tral**, from **Woodchuck Industries Inc.**, for the Apple II family. Introductory price $125.

Woodchuck Industries Inc., 340 W. 17th St. #2B, New York, N.Y. 10011, (212) 924-0576.

CIRCLE 107 ON READER SERVICE CARD

Pascal graphics

Turbo Pascal users can add drawing and animation to their programs with **Diversified Educational Enterprises Inc.'s Turbo Graphics**, priced at $39.95 for the IBM PC or compatible.

Diversified Educational Enterprises Inc., 725 Main St., Lafayette, Ind. 47901, (317) 742-2690.

CIRCLE 108 ON READER SERVICE CARD

Turbo screens, too

Turbo Pascal programmers can write and display full screen programs almost instantly with **Technisoft's FASTSCREEN**, for IBM PC and compatible, priced at $29.95.

Technisoft, 1710 Allied St., Ste. 37, Charlottesville, Va. 22901, (804) 979-6464.

CIRCLE 109 ON READER SERVICE CARD

Pascal to C conversions

IBM PC users can convert Apple, UCSD, MT+, Turbo, and MS-Pascal programs into equivalent C programs for $5,000 per site license, or use the translation service for $50/hour. **TGL Inc.** will convert 500 lines of Pascoul (Pascal) source free.

TGL Inc., 4400 Sulphur Springs Rd., Corvallis, Ore. 97330, (503) 745-7476.

CIRCLE 110 ON READER SERVICE CARD

New C libraries

Through **Greenleaf Software Inc.**, C programmers can add 230 functions to support Microsoft C or MS-DOS, supporting a variety of compilers on the IBM PC or compatible, for $185.

Greenleaf Software Inc., 1411 LeMay Drive, Ste. 101, Carrollton, Texas 75007, (214) 446-8641.

CIRCLE 111 ON READER SERVICE CARD

C Programmers:

Consider 104 Ways To Be More Productive

If you find and choose the right development software, you can: cut development effort, make impractical projects feasible, and eliminate unproductive, frustrating aspects of programming.

Confused? We'll help you sort thru the huge number of alternatives. Call for comparisons or information.

We carry 27 C Compilers, 4 C Intepreters, 49 Support Libraries, 5 C source debuggers, and 19 other C Addons for programming with MSDOS, Macintosh, or CP/M — more than 104 products, really. Here are some of the best products available:

Learn C Programming Only $95

"Introducing C" Interpreter

Computer Innovations has done it again! This interactive implementation is combined with a full screen editor and a thorough, self-paced manual.

You can develop programs faster by getting immediate feedback. Programs will start instantly upon your command. There is no need to wait for "compile and link."

Introducing C includes demo programs, powerful C language interpreter, complete C function library, full screen editor, color graphics, and C language compatibility. PCDOS $95

Simplify Screen Management

Windows for C

Keep your software up to date with the latest screen management features:

- Pop up menus and help files
- Instant screen changes
- Multiple windows
- Complete color control

Windows for C offers all of these plus much more in an integrated, compact, easy-to-use library of object code functions. Thorough, reference manual. Support for all memory models of popular C compilers. New version 3.1 offers enhanced portability and TopView compatibility. Full source available. MSDOS $180

Which Compiler Features Do You Need?

Optimizing C86 Compiler

Over the years the Optimizing C86 has evolved to be the most complete set of C compiler tools. It includes utilities, a rich library, and thorough tech support. In line 8087/287 routines run up to 100 times faster than the 8086 math package. The source code to all routines is included, so you have complete control over how they work. Thorough ROM support, Intel UDI & VMS cross versions are available.

More of the features you want include:

- special IBM-PC library • 2 math and 2 I/O libraries
- full memory utilization of the 8086/88/186/286
- compatibility with most commercial libraries
- object and source module librarian MSDOS $339

Get File Access with TIGHTER Control

db_VISTA Data Management

Full source, no royalties and "normal" indexed file management are part of db_VISTA. Get more for the price of only an ISAM.

You can minimize data stored and access records even faster and more logically than just using indexes. Example: address and transaction data should not require redundant storage of customer names or numbers. Use pointers. Related data fields point to other related groups — the "network model" of data.

Use db_VISTA as a "normal ISAM" or save programming time, access time and file size. Lattice, C86, Williams, Desmet, Microsoft C.

MSDOS. Source $450, Object $275.

SORT/MERGE Files for Clean, Fast Maintenance

with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added.

OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, FTN, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files Input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS. $90

Add Communications Features to Your Programs

Greenleaf Comm Library

Greenleaf now enables you to communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt driven system. This eliminates the extra function of separately calling up the communications program.

Included are 1 library/object files, 100 functions; 100 page manual, complete source code, library tailor-made to suit compiler and memory, Hayes-compatible modem commands, and a complete sample file transfer program. MSDOS $169

Inventive Programming Becomes Possible with 300 + ESSENTIAL, tested, fast, routines to Rely On.

C Utilities Library by Essential Software

Recent Enhancements to Graphics, Windows, AT Support

Every application you write is likely to require functions where you feel like you are "reinventing". Don't. Even if you use only 5% of this library, you will come out ahead on schedule and cost.

Full business Graphics, Window support, polled Communications, and Data Entry support have recently been added/upgraded along with more functions for DOS Interface and AT support. String handling, screen control, "word processor" functions, memory management, directory and path access, date handling, program chaining, keyboard and printer control are traditional strengths.

Full source code is included. No royalties are charged to include functions in your programs. 95% are C for portability and to make it practical for you to understand or modify them.

Lattice, Microsoft, C86, Mark Williams, Aztec, Desmet and Wizard C are supported. Specify which you need.

Substantial time, effort, testing and attention has been invested by Essential Software developing, documenting and supporting this comprehensive library. Make new projects practical and interesting. Use this tested and reliable library.

Some functions are PC-specific. Most support any MSDOS. $159.

Call for details, comparisons, or for our "C Extras Packet" with over 50 pages of information about C support products.

THE PROGRAMMER'S SHOP

The programmer's complete source for software, services and answers

128-LC Rockland Street, Hanover, MA 02339 (617) 826-7531 (800) 421-8006

Ask about COD and PO's. All formats available. Prices subject to change. Names of products and companies are generally their trademarks.

CIRCLE 78 ON READER SERVICE CARD

An industry look at 15 Forths

By Michael Ham, Michael McNeil, Stephen Martin, and William Lindow

In any review of Forth packages, the reviewer faces a frustration unknown to reviewers of languages such as BASIC and FORTRAN. Because Forth is an extensible language, its users can add new commands to it at any time—indeed, programming in Forth precisely consists of adding commands to address a specific task. Thus the lack of a particular command or feature in a Forth package must be measured against how difficult it would be for the user to add that command or feature.

In keeping with Forth's guiding philosophy of simplicity, most vendors avoid including nonessential commands as part of the package if the user can easily create them. Each command takes up room in the system and adds to the bulk of documentation. If the command clearly is needed by a large number of users, well and good; but if the command would be used only in special circumstances and is relatively easy to build, it probably is better to let the user add the command when (and if) it is needed. Thus the presence or lack of a particular command is not necessarily of any real significance.

So on what basis does a potential user choose a Forth?

One criterion is speed. The top of the Forth dictionary consists of words written in Forth—the same sort of extensions to the language that the user provides. But the very innermost words within any Forth dialect are written in machine code. Their structure and polish to some extent determine how fast the Forth will be at traditional benchmark tasks. The more words written in machine code, the faster (and less portable) the particular Forth system is likely to be.

It is worth pointing out, however, that the only true benchmark of performance is a job mix of tasks specific to the user's own applications. And because one of those tasks is programming, overall speed should also be measured from the inception of the idea to the arrival from the completed program of real output—that is, real-world speed concerns not only internal performance once the program is complete but also the ease with which one can program in a particular Forth.

Thus a second key criterion for language selection is the quality of the documentation. Is it clear, comprehensive, and readable? And for those instances in which a user cannot make out what is meant, is there telephone support? Is the vendor available to answer specific questions?

Another factor affecting the ease of using the Forth is the richness of the set of extensions provided by the vendor. For example, considerations of speed suggest that in an application program the developer may want to write some parts of it in machine code. Forth does not use a separate assembler and linker, the procedure is to begin the definition with *CODE* instead of *:* and then enter the name of the word followed by its definition written in assembly language instead of Forth. The definition ends with *ENDCODE* instead of a semicolon.

The usual development procedure for speed-critical applications is to get the entire program running in high-level Forth and then systematically replace the most frequently encountered Forth definitions with assembler definitions. Since 95% of a program's time typically is spent within 5% of the code, this procedure allows the programmer to gain assembly language speed with very little assembly language programming. Users whose applications demand extremely fast processing speeds will want a Forth that includes an assembler. (It is worth pointing out that most Forths are fast enough for many users to write all their code in Forth without having to dip into assembly language at all.)

The Forths reviewed here generally include an assembler and an editor. Look also for other extensions that might be important to you—a special capability in graphics might be something you prize, or a need to work with files generated within a particular operating system.

Another important criterion is vendor support, which is expressed in several ways. One important kind of support is the fixing of bugs. Forth systems, particularly early in their evolution, are like any other programs: they exhibit occasional bugs that arise only in particularly obscure encounters with the operating environment. If the vendor is responsive to reports of such problems, the vendor's Forth will quickly become reliable through its trial by fire in real-world encounters. This evolution presupposes an ongoing effort on the part of the vendor.

An aspect of vendor support already touched on is in the number of extensions provided to the basic Forth package. Allied with this kind of support is the degree to which the vendor responds to customer requests and suggestions. Some vendors are extremely open to ideas for extensions and improvements, which is reflected in the polish their systems exhibit. For example, the editor in PC/Forth v. 2.0 used its own unique set of control-key commands. Acting on a user suggestion, Laboratory Microsystems revised the editor in v. 3.0 and installed WordStar control-key commands, to the great benefit of users accustomed to those commands.

Some vendors also act as a clearinghouse for user-developed software and suggestions. The electronic bulletin board systems installed by those vendors not only carry answers to technical questions but also allow users to upload and download source code of various routines, written in the vendor's Forth.

Another consideration, particularly if you plan to develop programs across a variety of computers, is the degree to which the vendor provides the same Forth on different machines. It obviously helps the developer tremendously if the bulk of the source code can be ported directly to another machine.

Developers are also apt to be keenly interested in the royalty fee and licensing arrangements for programs written in a particular Forth. Indeed, anyone who writes a program that may have more than one user should check the conditions under which the program can be shared. Some of the licensing agreements require the payment of a significant licensing fee even if you merely want to share with a local users group some program you developed. (Of course, it is perfectly legitimate to pay no license fee or royalty if you sell or give your program to those who already own that Forth, since they can use it on their own systems.)

It should be noted that many system features may seem positive or negative according to the individual's point of view. To the experienced Forth programmer, lack of long memory addressing operators, for example, is not particularly a liability for a given system. When needed, the individual can add operators such as these to the language in five minutes or so, including testing. In the mean time, the programmer appreciates the lean and trim nature of polyFORTH, say, which provides a larger programming space than its small-model memory addressing might lead one to imagine.

Many experienced Forth programmers—fluent in Forth and not really wanting or expecting to deal with other environments—will value the approach taken by polyFORTH and other systems where a common non-DOS Forth environment is created and maintained across a number of different processors. On the other hand, the individual who desires to coexist along with non-Forth programs in a shared DOS data environment will find this approach to be a handicap.

Although all these systems claim to follow some Forth standard or other, modifications to the benchmark programs were necessary in almost every case, mostly to semistandardized words such as 2DUP, 2*, etc. The file interface varied

The community of Forth users

Forth has come up from underground. The language was conceived and developed on the job, far from the normal programming language incubation site of academia or research and development institutes.

Charles Moore stated that his goal in developing the language was simply to make himself a more productive programmer (see *COMPUTER LANGUAGE*'s ComputerVisions, Premier issue, pp. 69-70, and Designer's Debate, March issue, pp. 19-24). He developed Forth gradually, over a period of several years in various programming jobs for underfunded enterprises that were trying to wring the maximum of performance out of the minimum of machines. Forth's simplicity, compactness, speed, and extensibility resulted from its development in the rough-and-tumble world of real-life applications.

As rumors about Forth's capabilities began to percolate throughout the microcomputer community, those intrigued were frustrated by the lack of access to any packages. FORTH Inc. was already in existence, but the cost of its system made it prohibitive to hobbyists who just wanted to learn the language. In the late 1970s, the Forth Interest Group, led by Bill Ragsdale, Kim Harris, and others, developed a public domain version of Forth for a variety of microprocessors and established an organization to bring together hobbyists and developers interested in the language. Building on the fig-Forth effort, a number of commercial vendors added polish, extensions, and support to the package, leading to today's relatively rich Forth environment.

Because Forth is a simple language to port, it is frequently the first language up on a new machine (as MacForth was on the Macintosh). And because Forth has so much power despite its simplicity, a variety of companies and products place their faith in Forth. Simon & Schuster's Typing Tutor III, Scarborough Systems' MasterType, a variety of programs from Electronic Arts, and many small dedicated devices, such as pocket translators and embedded control chips, all do their magic via Forth.

The Forth Interest Group continues to act as the main national Forth clearinghouse of information and activity. Membership is $20 per year and includes a subscription to the bimonthly *Forth Dimensions*. (Forth Interest Group, P.O. Box 8231, San Jose, Calif. 95155, (408) 277-0668.) Every fall the Forth Interest Group has a convention of Forth users and vendors. The convention this year will be September 20 and 21 in Palo Alto, Calif.

Another publication of interest to Forth programmers is *The Journal of Forth Application and Research*, published quarterly by The Institute for Applied Forth Research Inc., 70 Elmwood Avenue, Rochester, N.Y. 14611.

Books on Forth are available from fig and from many of the Forth vendors reviewed in this issue. The standard beginner's book is *Starting Forth* by Leo Brodie. Other good introductions are *The Complete Forth* by Alan Winfield and *Mastering Forth* by Anita Anderson and Martin Tracy. Intermediate texts include *Thinking Forth* by Leo Brodie and *Forth Tools and Applications* by Gary Feierbach.

C

HOW EASY DEVELOPING PROGRAMS CAN BE WITH THE C UTILITY LIBRARY!

COMPLETE. Over 300 tested and well documented functions. All features of the PC &AT are at your finger tips.

ADVANCED. Features like windows, data entry fields, switching displays, and batch file execution are supported.

GRAPHICS. A complete set of fast business graphics functions. Our low level graphics routines are the fastest in the business.

SOURCE. All source code is included. 95% of the library is written in C. Only functions demanding fast execution speed are in Assembler.

COMPATIBLE. With Lattice, Microsoft(3.0), CI-C86, Aztec, DeSmet, Wizard, and Mark Williams. All memory models are supported.

MUCH MORE. Dozens of string functions, the best time & date math and formatting, DOS directory and file mgmt., keyboard control, polled async communications, and more. **NO ROYALTIES.** We **SUPPORT** what we sell.

C UTILITY LIBRARY - $185
COMPILERS: Lattice $349, CI-C86 $329, Mark Williams $449. **Save $40-$50** with compiler & library package.

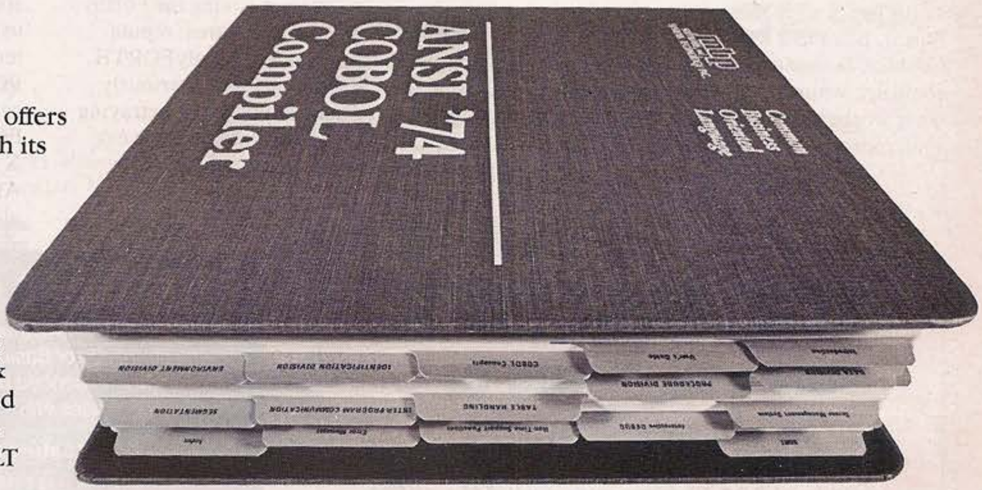
Specify compiler and version number when ordering. Add $4 for UPS or $7 for UPS 2-day shipping. NJ residents add 6% sales tax. VISA, MC, Chks & qualified PO's.

ESI
Essential Software, Inc.
P.O. Box 1003
Maplewood, New Jersey 07040
914/762-6605

CIRCLE 33 ON READER SERVICE CARD

mbp COBOL
for your IBM/PC

The new standard for convenience.



Now, the mbp COBOL Compiler offers unrivaled convenience to go with its unmatched performance.

Here are the convenience features you've wished for:

1) an enhanced Screen Management System with program-controlled video attributes and color; 2) support for PATH & sub-directories; 3) DOS command execution from within a COBOL program; 4) 'permanent' DEFAULT modification.

The new mbp Compiler has them all! And they're exclusives: you get them *only* with mbp.

Plus, it's 4 times faster.

Because the mbp COBOL Compiler generates native machine language object code, it executes programs *at least* 4 times faster (see chart). Now, we've made that performance even more convenient to use.

GIBSON MIX Benchmark Results
Calculated S-Profile
(Representative COBOL statement mix)
Execution time ratio

mbp* COBOL	Level II* COBOL	R-M* COBOL	Microsoft* COBOL
1.00	4.08	5.98	6.18

*128K system with hard disk required. IBM PC & AT are IBM TMs; Netware is a Novell, Inc TM; Level II is a Micro Focus TM; R-M is a Ryan-McFarland TM, Microsoft is a Microsoft TM.

The complete COBOL.

An Interactive Symbolic Debug Package included standard; Multi-keyed ISAM structure; SORT & CHAIN; GSA certification to ANSI '74 Level II; IBM/PC-AT and TI Professional compatibility; with mbp, you get it all. Optional: Novell NetWare interface.

mbp COBOL: the choice of professionals.

It's no surprise more and more companies like Bechtel, Bank of America, Chase, Citicorp, Connecticut Mutual, Hughes Aircraft, McDonnell-Douglass, and Price-Waterhouse choose mbp COBOL.

Make it your choice, too. Just send the coupon, or call, for complete information. Today.

mbp COBOL. $1000

Please send complete mbp COBOL information to:

NAME ____________________

COMPANY ____________________

ADDRESS ____________________

CITY/STATE/ZIP ____________________

PHONE ____________________

mbp Software & Systems Technology, Inc.
7700 Edgewater Drive, Suite 360
Oakland, CA 94621
Phone 415/632-1555

mbp®

CIRCLE 53 ON READER SERVICE CARD

drastically, with phrases opening the benchmark source file ranging from OPEN BNCHMARK.FTH to #0 FCB BNCHMARK SCREEN-OPEN to "BNCHMARK.SCR" SCREENS OPEN IS DEFAULT.

Finally, it should be pointed out that most of the Forths reviewed here are in a state of continuing development and refinement. If you are interested in any particular packages, you should contact the vendor to get current information.

IBM PC FORTHS

The IBM PC has attracted the attention of numerous Forth vendors who offer a broad line of products tuned to take advantage of the various special features of the IBM and compatible products. This review discusses the capabilities of IBM Forths from 11 vendors, in alphabetical order by vendor name.

FORTH Inc. polyFORTH II

FORTH Inc. was founded in the early 1970s by Charles Moore (the initiator of Forth), Elizabeth Rather, and Edward Conklin. Rather and Conklin still run the company, while Moore has gone on to other work, including the development of a microprocessor that runs Forth as its native language. From the beginning, FORTH Inc. has emphasized simplicity, compactness, and speed, avoiding anything that might be taken as a frill. The company's product philosophy could be summarized as lean and mean.

FORTH Inc. ported its Forth to processor after processor until polyFORTH systems today run on Intel 8080, 8086, DEC PDP/LSI-11, Motorola 6809 and 68000, RCA 1802-1805, NCR-32, and NC-4000 processors, FORTH Inc. also does a large amount of contract programming work in polyFORTH, specializing in jumping into software engineering projects in a state of crisis, replacing the existing operating systems and applications with customized polyFORTH, and using the speed and compactness of its Forth (along with its programmers' familiarity with it) to save the day.

The polyFORTH systems all include multitasking. They are multiuser when desired and fast-compiling and compact even by Forth standards. polyFORTH commands a high price among the Forths reviewed, based on its features, reputation, and performance. Yet polyFORTH on the IBM PC presents some curiously old-fashioned and creaky traits, betraying its age and history and perhaps the weak points in a product philosophy of spareness.

Forth would not be the language it is if Moore had not insisted on going his own way and holding fast to simplicity and compactness as guiding principles. However, in interpreting this overall strategy, FORTH Inc. has nearly always found that the simplest approach for any given implementation is to seize the entire machine and use Forth as its own operating system. Even when a computer has I/O routines in ROM, FORTH Inc. generally prefers to interface directly with machine hardware. While this approach does provide a simple and compact structure for the operating environment and squeezes amazing capabilities and speed into small machine computers, it also has the drawback that the resulting programs are incompatible with data produced by other (nonForth) programs and cannot run in a common environment with other programs the users may have.

This kind of simplicity has also led to compatibility problems as the hardware has evolved. While other programs were somewhat insulated from the hardware by using the manufacturer's operating system, the native polyFORTH for the IBM PC could not run on the XT when the XT came out. Then the version of polyFORTH that had been modified for the XT would not run on the PCjr or the AT. Another example: polyFORTH's standalone interface to the DOS file system,

THE HAMMER

Software Tools in C

More than just BIOS/DOS access, THE HAMMER Library also provides routines for multi-level **123-like menus,** easy **data entry & verification** of strings, numbers, & dates, screen attribute control, date & time processing, AND MORE.

Super data entry routines give programmer a natural, strong interface with the user. They work in both "single-field" and "multi-field" **(full screen editing)** modes. This is NOT just another library of general purpose routines.

For: IBM/PC family with DOS 2.00+
C Compilers: CI-C86, Lattice, DeSmet, and new Microsoft V3.00

$195 with source code and manual. To order or inquire, **CALL OR WRITE:**

O.E.S. SYSTEMS
1906 Brushcliff Road
Pittsburgh, PA 15221
412/243-7365

Looking for the right tool for the job?
GET THE HAMMER

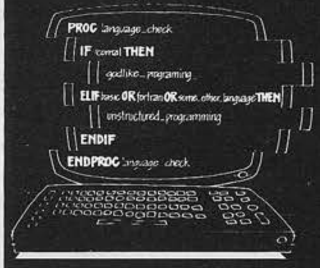
CIRCLE 98 ON READER SERVICE CARD

The best of two worlds

The MYTECH COMAL interactiv programming language gives you the simplicity of Basic and the power of Pascal. Many of the concepts are influenced by ADA®¹, for example exception handling, packages etc.

Mytech Comal features

- Friendly, interactive user interface with help facilities
- 100% orthogonal.



```
PROC language_check
  IF comal THEN
    godlike_programing
  ELIF basic OR fortran OR some_other_language THEN
    unstructured_programming
  ENDIF
ENDPROC language_check
```

- Screen editor
- Fulfills the Comal 2.00 requirements
- Available for IBM®^{2}, MACINTOSH®^{8}, HP, OLIVETTI, APRICOT®^{9}, WICAT etc
- Implemented on UNIX®^{2}, CPM/86®^{3}, C-CPM/86®^{4}, MS-DOS®^{5}, PC-DOS®^{6}
- Easily ported to 16/32-bits systems
- The package concept makes Mytech Comal extensible
- Turtlegraphics package (LOGO) is available
- Easily customized for foreign languages
- Support for the 8087.
- Is written in "C".
- Comal is an official education language in Europe.

Mytech Comal *is the perfect language for students as well as for the professional programmer. For further information please ask for a data sheet. Or why not do it the right way, order your* ***Mytech Comal*** *system today and move in to a new fascinating and powerful programming dimension.*

1) DoD, 2) IBM, 3) Digital Equipement, 4) AT & T, 5&6) Digital Research, 7) Microsoft, 8) Apple Computer Inc., 9) ACT Int.

MYTECH DATA AB

"Multum in parvo"

Mytech Software INC, 111 20 Rosell Street no.E, San Diego, CA 92121 USA
Tel. 619-452-9847, Telex. 705340

CIRCLE 19 ON READER SERVICE CARD

WIZARD C

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

- All UNIX System V language features
- Support for 8087, 80186 and 80286
- Full library source code included
- Cross-file checks (full UNIX lint)
- Uses MS-LINK or PLINK 86
- ROMable data options
- In-line assembly language
- Cross compilers available
- Third party software available, including PANEL

The new standard for C Compilers on MSDOS!

Only $450.

(617) 641-2379

WIZARD
Systems Software, Inc.

11 Willow Court
Arlington, MA 02174

VISA
MasterCard

CIRCLE 86 ON READER SERVICE CARD

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K& RC. Source code included, No Royalties, Powerful & Portable.

BTree Library **75.00**

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver **40.00**

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make **59.00**

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

For more information call or write:

softfocus

Credit cards accepted.

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903
(416) 844-2610

Dealer inquiries invited.

Chuck Moore's Forth Chip.
Now Available.

Novix

Beta-board

A single-board computer designed for your evaluation of the Novix NC4000P® Forth microprocessor in real-world applications.

For Complete
Information Contact:
Novix
10590 N. Tantau Ave.
Cupertino, CA 95014
408 / 996-9363

CIRCLE 73 ON READER SERVICE CARD

designed for DOS 1.x, could not be used to access DOS 2.x files, even on disks lacking a subdirectory structure. Thus the FORTH Inc. approach forces Forth to be agile in dancing to the hardware manufacturer's tune and tempo. polyFORTH also is not compatible with the Forth-83 Standard; the system is essentially Forth-79 Standard.

The problems in compatibility and marketing are slowly moving FORTH Inc. toward safer methods of interfacing to its computers. Two level 3 polyFORTHs are reviewed here; one is the traditional stand-alone polyFORTH for the IBM PC and performs its I/O by direct interface to the hardware.

The other polyFORTH system reviewed is even more integrated into the PC environment because it resides in an MS-DOS COM file and performs its I/O via DOS calls. Thus this system presumably is capable of running on any MS-DOS computer. Even this polyFORTH, however, displays minimal integration into the DOS environment and remains much closer to the polyFORTH world than the DOS world. For its disk I/O, MS-DOS polyFORTH simply accesses an image of the native polyFORTH system disk that has been transferred, block by block, into a DOS file. polyFORTH still does not seem to be really at home under MS-DOS.

FORTH Inc. says that the fully native polyFORTH for the IBM PC will be continued. A soon-to-appear upgrade of the native product will include an expanded (stand-alone) DOS file-handling package with the ability to create a DOS-bootable COM file containing a native application. Accompanying current versions of polyFORTH are various utilities, such as an 8086 assembler, command-type block editor, and polyFORTH's own file system and data base manager, which allow files and record structures to be declared and manipulated.

Developers should note that FORTH Inc. requires no royalties or license fees for program products written in polyFORTH, provided that the end user has no access to the underlying Forth. If the user can access the underlying Forth (under any circumstances), a fee is required. For serious development work, FORTH Inc. sells a level 4 version of the product (at $3,200) that includes the complete system source code and a target compiler that allows the customer to tailor the complete system to his or her own situation. The system, consistent with its minimalist philosophy, does not have any tools for creating binary (precompiled) overlays. An easy-to-use command is included for creating bootable and precompiled programs.

polyFORTH documentation is very complete. Two large manuals that come with the polyFORTH system are the 432-page *polyFORTH II Reference Manual* and a 91-page *Intel 8086 CPU Supplement* to the reference manual. Also included is Leo Brodie's *Starting FORTH*, an excellent introduction to Forth in general and polyFORTH in particular. Commissioned in 1981 by FORTH Inc., *Starting FORTH* has become a Forth standard in its own right.

Harvard Softworks HS/FORTH

Harvard Softworks' name reflects its Massachusetts origins, though the company is currently located in Ohio. Its HS/FORTH (v. 2.05) includes several features unusual among the other Forths in this review. HS/FORTH as booted is a Forth-79 Standard system, but it includes a Forth-83 Standard compatibility mode under which it conforms to the Forth-83 Standard. HS/FORTH runs on various machines that run MS-DOS since its standard system I/O goes through DOS.

The HS/FORTH system is moderately integrated into the MS-DOS environment. The program is stored as a DOS COM file and allows access to files via the mechanism of DOS 1.0 file control blocks. Not only is HS/FORTH capable of inter-

Now for the
IBM PC, XENIX, UNIX, VMS...

WINDOWS FOR C™

Advanced Screen Management
Made Easy
A video tool kit for all screen tasks

- Pop-up menus and help files
- Unlimited files and windows
- Rapid screen changes
- Logical video attributes
- Complete color control
- Horizontal and vertical scrolling
- Word wrap
- Highlighting
- Plus a library of over 65 building block subroutines

So easy to learn and easy to use,
you'll wonder how you ever managed
without it.

Provides application portability between the IBM PC and XENIX, UNIX, VMS or any terminal-based system.

Full support for IBM PC/XT/AT and compatibles; Lattice C, C1-C86, Mark Wm. C, Aztec C, Microsoft C, DeSmet C (PC/MSDOS); PC/XENIX. Source version available for Unix and other OS.

NEW Ver. 4.0
Logical attributes
Easier menus
New pop-up functions
WINDOWS FOR C
PCDOS
(specify compiler) **$245**
PC/XENIX $495
UNIX and other OS Call
Full Source Available

Vermont Creative Software

21 Elm Ave.
Richford, VT 05476
802-848-7738, ext. 41,

Master Card & Visa Accepted
Shipping $2.50
VT residents add 4% tax

Trademarks - UNIX, AT&T; XENIX, Microsoft; VMS, DEC.

CIRCLE 8 ON READER SERVICE CARD

LISP.
OUR BUSINESS IS LISP.
PERIOD.

The LispLab is a complete Lisp resource. We not only sell Lisp, we know Lisp. If you are looking for a Lisp implementation for your computer, you should talk to us. We stock a complete selection of interpreters and compilers for most computers and operating systems including MS-DOS, UNIX, CP/M, VAX/VMS, RSX-11M, OS-9 and others. We can help you make your selection, get you up and running, keep you informed of updates and save you money in the bargain.

LispLab pricing is as good or better than the mail order program shops, yet we provide a depth of knowledge and range of services they cannot match when it comes to Lisp. Please inquire about our schedule of seminars and on-site training services. When we sell you a Lisp, we intend to make you productive in Lisp.

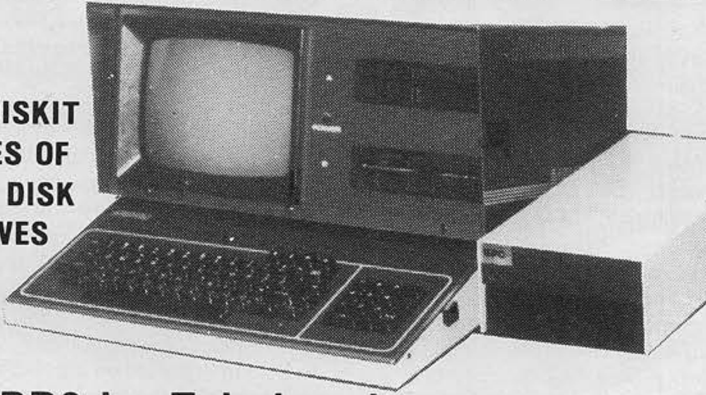
Our bi-monthly publication, The LispLetter, is full of news, information, updates, code and bug reports. A sample copy is yours for the asking. Customers will get a free one year subsciption. Others may subscribe for eight dollars per year.

If your plans include Lisp, they should also include LispLab. Please write or call.

LispLab
Box 548
Windham, New Hampshire 03087
Ph. (603) 898-4924

CIRCLE 58 ON READER SERVICE CARD

FOR THE SERIOUS KAYPRO® USER



THE DISKIT
SERIES OF
HARD DISK
DRIVES

now
with

ZCPR3 by Echelon, Inc.

Now you can add from 5 to 40 Megabytes of fast-access Winchester storage to your KAYPRO 2, 4, or 10. The DISKIT is only 4 inches high; 5.7 if you get the two drive model with the *removable* 5 or 10 Mb. cartridge, and weighs less than 10 pounds. Easily disconnect DISKIT from the computer whenever you want, and if more capacity is required, just swap your drive for a larger model.

Our DISKIT Model 10 has 10.8 Megabytes of *formatted* capacity . . . 20% more than a Kaypro 10, and runs about twice as fast. Installs in minutes.

Call SPC now and ask for more information. Quantity and prepayment discounts are available.

SYSTEMS PERIPHERALS CONSULTANTS
9747 Business Park Avenue
San Diego, CA 92131
(619) 693-8611

CIRCLE 38 ON READER SERVICE CARD

YOU NEED A GOOD C LIBRARY

C POWER PACKS™

COMPLETE SOURCES
NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."
— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."
— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." — Microsystems, Oct. 1984

■ **PACK 1: Building Blocks I** $149
DOS, Keyboard, File, Printer, Video, Async

■ **PACK 2: Database** $399
B-Tree, Virtual Memory, Lists, Variable Records

■ **PACK 3: Communications** $149
Smartmodem™, Xon/Xoff, X-Modem, Modem-7

■ **PACK 4: Building Blocks II** $149
Dates, Textwindows, Menus, Data Compression, Graphics

■ **PACK 5: Mathematics I** $99
Log, Trig, Random, Std Deviation

■ **PACK 6: Utilities I (EXE files)** $99
Arc, Diff, Replace, Scan, Wipe

Master Card/Visa, $7 Shipping, Mass. Sales Tax 5%
ASK FOR FREE DEMO DISKETTE

NOVUM ORGANUM INC.
SOFTWARE HORIZONS INC.

165 Bedford St., Burlington, MA 01803
(617) 273-4711

CIRCLE 25 ON READER SERVICE CARD

C

SOFTWARE
DEVELOPERS!

V - FILE
THE VIRTUAL MEMORY
FILE MANAGER

Let V-FILE save precious development time & cost as you create efficient applications with the power of VIRTUAL MEMORY.

DON'T RE-INVENT THE WHEEL

Why spend weeks or months coding and debugging file and memory management systems when you can order V-FILE today. V-FILE is a library that you can link with your code to provide sophisticated virtual file and memory management — allowing you to concentrate on developing your application.

VIRTUAL DATA
OBJECTS SUPPORTED!

Data is referenced by using VIRTUAL MEMORY DATA HANDLES. Your code doesn't need to know whether the data is actually on disk or in RAM. Swapping between disk and RAM and updating files on disk is handled automatically and transparently! Complex VIRTUAL DATA STRUCTURES can be created by linking with data handles instead of pointers.

CHECK THESE FEATURES!

- Multiple, independent swap buffers
- Multiple files per swap buffer
- Highly efficient swap algorithm
- Automatic file updating
- Data prefetching supported
- Data may be locked in memory
- Memory buffers may be flushed
- Makes full use of extended memory on IBM PC/AT
- SOURCE CODE AVAILABLE
- NO ROYALTIES REQUIRED

Supports Dos 2.00+ with
Lattice & Microsoft C compilers
Supports Microsoft windows

$299

V-FILE TM

Contact:
MindBank, Inc.
4620 Henry Street
Pittsburgh, PA 15213
412/683-9800

VISA/MASTER CARD ACCEPTED

CIRCLE 63 ON READER SERVICE CARD

preting text files, all the system source is distributed in that format. Optional packages allow interpretation of files and non-DOS disks containing Forth blocks. HS/FORTH does not support DOS 2.x path names.

HS/FORTH possesses a highly segmented memory organization, allowing it to take advantage of the 8088 processor's method of memory management. Stacks, tasks, high-level definitions, machine code, vocabulary word headers—everything is placed in a separate memory segment. Thus headers can be easily discarded by simply using a *BEHEAD* command, giving it either a single word or a range of words; this provision frees up memory for bigger applications.

HS/FORTH is accompanied by various utilities: assemblers (which include syntax checking) for the 8088, 80186, and 8087 processors, an editor, and a program development toolkit. The system includes an example of multitasking, but multitasking is not built in. IBM features, such as a speaker, are supported through special commands; for example, half a dozen commands control sound.

Harvard Softworks requires no royalties or fees for packages written in HS/FORTH that are distributed as bootable COM files if they do not contain the HS/FORTH definition lists (which are easily removed with the *BEHEAD* command). If the application allows the end user to enter commands to the interpreter (even though the user cannot define new commands), the royalty is $1 per copy, with a maximum of $500 per program product.

SOTA: A CP/M Forth

In addition to the recently released SOTA CP/M Forth, CP/M Forth products are also available from Laboratory Microsystems Inc., MicroMotion, and Unified Software Systems.

SOTA Computing Systems is the only company to bring out a new CP/M product: a fig-Forth model. fig-Forth antedates even the Forth-79 Standard, and most vendors have upgraded from this early system. The company plans publication in September of a Forth that can be easily configured to conform to fig-Forth, Forth-79 Standard, or Forth-83 Standard.

SOTA is primarily concerned with the implementation of various interpreters and compilers on mainframe computers. This Forth is SOTA's first experience with the retail and mail order market. If the manuals are any guide, I believe SOTA will be successful.

The SOTA manual set contains five sections. They are all printed on a dot-matrix printer, probably because they use many icons. One problem with the dot-matrix print is the striking resemblance between a boldface slashed-zero and a black blob. Each section has a table of contents, but none are indexed.

The first manual is the six-page fig-Forth *Installation Guide*. It provides a step-by-step procedure for installing fig-Forth with a detailed description of what is happening at each step. Icons are used where necessary for clarity.

The *Users Manual* describes the vocabularies for Forth execution, Forth compilation, the editor, and the Z80 assembler. In addition, the system messages and disk error messages are described. Each word is accompanied by an icon that specifies whether the word is compile mode only, a defining word, an immediate word, a run-time routine, and/or a SOTA extension. In addition, the pronunciation of the word is given (of some importance since some words contain symbols or consist of symbols), and its effects and any restrictions on its use are described. The Z80 assembler vocabulary has many examples to clarify the context of each word. This manual is 88 pages.

Brief tutorials are provided for the editor and for the assembler. The editor tutorial is only three pages long and contains only one example. The screen editor commands and the mechanics of the editor operations are briefly discussed.

The assembler tutorial is designed for people already familiar with Z80 assembly language programming. It contains references to a Forth tutorial manual to be released in September. Even though it is only eight pages long, it does contain many good examples.

The *System Guide* gives very detailed information on interfacing I/O devices to TRS-80 models 1, 3, and 4. It also discusses the video screen, printer, keyboard, and disk driver.

If SOTA's planned documentation efforts for a beginner's tutorial and a programmer's guide are as good as these manuals, then this may be the best documented CP/M Forth system. SOTA's licensing agreement allows the developer to distribute programs written in this Forth with no payment of royalties or license fees, provided the end user has no access to the underlying Forth.

The vendor offers telephone support and plans a bulletin board system for the fall.

Now Supports Microsoft 3.0

$149.95

Vitamin C

Windows that open, close, grow, shrink, move & scroll. Input validation,formatting, editing & processing. Help messages by field, by key word & from help file. Date & time math, attribute control, pull down menus, and more!

Now even your simplest applications can easily include features that you didn't used to have time or patience enough to tackle. Vitamin C is more than a library full of building blocks. It is a well planned, tightly woven set of high level functions for quick results PLUS low level routines for complete control. Complete source code, no royalties. Great manual with tutorial and reference. Sample programs too! For Microsoft 3.0, CI-C86, & Lattice C. Call for other versions, systems & products.

CALL TODAY! Or send 149.95+ $3 shipping and handling. Texas add sales tax. MasterCard and Visa accepted.

Creative Programming
Box 112097
Carrollton, Tx 75011
(214)243-6197

Extended Technical support availab le.

CIRCLE 52 ON READER SERVICE CARD

A FULL C COMPILER FOR $49.95

The Eco-C88 C compiler is setting a new standard for price and performance. Compare Eco-C88's performance to compilers costing up to 10 times as much:

	Seive	Fib	Defref	Matrix
Execute	12.1 sec.	43.1 sec.	13.7 sec.	21.3 sec.
Code Size	7782	7754	7772	9120
Compile-link	76 Sec.	77 Sec.	77 Sec.	92 Sec.

Eco-C88 Rel. 2.20, on IBM PC with 2 floppy disks, 256K. Benchmarks from Feb., 1985 **Computer Language.**

Eco-C88 includes:
- ⋆ All operators and data types (except bit fields)
- ⋆ Error messages in English with page numbers that reference the **C Programming Guide** – a real plus if you're just getting started in C.
- ⋆ Over 170 library functions, including color and transcendentals
- ⋆ New Library functions for treating memory as a file
- ⋆ User-selectable ASM or OBJ output (no assembler required)
- ⋆ 8087 support with 8087 sensed at *runtime*
- ⋆ cc and "mini-make" for easy compiles (with source)
- ⋆ Fast, efficient code for all IBM-PC, XT, AT and compatibles using MSDOS 2.1 or later.
- ⋆ Complete user's manual

If ordered with the compiler, the C library source code (excluding transcendentals) is $10.00 and the ISAM file handler (as published in the **C Programmer's Library,** Que Corp) in OBJ format is an additional $15.00. Please add $4.00 for shipping and handling. To order, call or write:

MasterCard VISA AMERICAN EXPRESS

Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220
(317) 255-6476 • 8:30–4:30

ES ECOSOFT INC.

Eco-C (Ecosoft), MSDOS (Microsoft), UNIX (Bell Labs), CP/M (Digital Research), Z80 (Zilog), 8086, 8087, 8088 (Intel).

CIRCLE 9 ON READER SERVICE CARD

C

POWER
C LIBRARIES
C WINDOWS

Best You Can Get!
325 Fully Tested Functions

Best Documentation
(over 400 pages)

SIX C LIBRARIES
For IBM PC, XT, AT

All Source Code. No royalties.
51 screen handling/graphics
50 cursor/keyboard/data entry
85 superior string handling
25 system status & control
72 utility/DOS/BIOS/time/date
42 printer control

RICHLY COMMENTED
EASY TO LEARN
EASY TO MODIFY

NO MATTER WHAT ELSE YOU HAVE
GET THESE!!

ALL 6 LIBRARIES $99.95
(ON FOUR DISKETTES)

POWER WINDOWS

PROFESSIONAL WINDOW MANAGEMENT

OVERLAYS, BORDERS, POPUP-MENUS, COLOR HIGHLIGHTING, HELP WINDOWS, STATUS-LINE, MONOCHROME OR COLOR, FILE, CURSOR, KEYBOARD CONTROL AND MORE !!

C WINDOWS: COMPLETE SOURCE CODE $99.95

ALL LIBRARIES PLUS WINDOW $159.95

Entelekon
SOFTWARE SYSTEMS
ENTELEKON 12118 KIMBERLEY
HOUSTON, TX. 77024 (713)-468-4412
VISA•MASTERCARD•CHECK

CIRCLE 40 ON READER SERVICE CARD

TURBO
Power

PROGRAMMER'S UTILITIES
especially for Turbo Pascal on
IBM PC/XT/AT and compatibles

MORE POWERFUL THAN UNIX UTILITIES!!!

These Ready-to-Use programs fully support Turbo Pascal versions 2.0 and 3.0, and PCDOS 2.X and 3.0. Here's what you get:

Pretty Printer
Standardize capitalization, indentation, and spacing of source code. Don't waste your own time! Several adjustable parameters to suit your tastes (works with any standard Pascal source).

Program Structure Analyzer
Find subtle problems the compiler doesn't: uninitialized and unused variables, modified value parameters, "sneaky" variable modification, redefined standard identifiers. Also generates a complete variable cross reference and an execution hierarchy diagram. Interactive or write to file (works with any standard Pascal source).

Execution Timer
Obtain a summary of time spent in each procedure and function of your program, accurate to within 200 microseconds. Also counts number of calls to each subprogram. Fully automatic.

Execution Profiler
Obtain a graphic profile of where your program spends its time. Interactive, easy-to-use. Identify weak code at the instruction level. (Profiler and Timer for Turbo Pascal Source code only.)

Command Repeater
Customize any operation by reading and parsing the standard input. Send up to 255 keystrokes to any executed program. Automatically generate DOS batch files.

Pattern Replacer
Find and REPLACE versatile regular expression patterns in any text file. Supports generalized wildcards, nesting, alternation, tagged words and more. Over a dozen programmer's applications included.

Difference Finder
Find differences between two text files, and optionally create an EDLIN script which rebuilds one from the other. Disregard white space, case, arbitrary characters and Pascal comments if desired.

Super Directory
Replace PCDOS DIR command with extended pattern matching, sort capability, hidden file display, date filtering, and more.

File Finder
Locate files anywhere in the subdirectory tree and access them with a single keystroke. Display the subdirectory tree graphically.

**AVAILABLE IN SOURCE
AND EXECUTABLE FORMAT**

Executable: $55 COMPLETE including tax and shipping. Compiled and ready to run, includes 140-page printed user manual, reference card and one 5¼" DSDD disk. Ideal for programmers not using Turbo. NOT copy protected.

Source: $95 COMPLETE including tax and shipping. Includes all of the above, and two additional DSDD disks. Disks include complete Turbo Pascal source code, detailed programmer's manual (on disk) and several bonus utilities. Requires Turbo Pascal 2.0 or 3.0.

Requirements: PCDOS 2.X or 3.0, 192K RAM — programs run in less RAM with reduced capacity. Two drives or hard disk recommended.

TO ORDER:
VISA/MasterCard orders, call 7 days toll-free 1-800-538-8157 x830. In California, call 1-800-672-3470 x830 any day. Or mail check/money order to:

TurboPower Software
478 W. Hamilton Ave., Suite 196
Campbell, CA 95008

For technical questions, call 408-378-3672

CIRCLE 57 ON READER SERVICE CARD

The HS/FORTH manual is 300 pages, with updates mailed to registered users. Harvard Softworks offers telephone support during the business day.

Laboratory Microsystems PC/Forth

Laboratory Microsystems Inc., a long-term vendor of high-quality Forth products, has released version 3.0 of its PC/Forth, a Forth-83 Standard system. Version 3.1 was about to appear at press time. PC/Forth runs on the IBM PC and AT, with versions also available for the HP-150 Touchscreen, the HP-110 Portable, and the Wang Professional. These various systems use the same command set but are hardware dependent for maximum performance.

PC/Forth is well integrated into the MS-DOS environment. All disk I/O uses DOS files and source code is stored in files, either in the traditional Forth format of 1K blocks or as text files that are loaded with the *INCLUDE* command. Text can be in ASCII files or even in some word processor formats (such as WordStar document files).

The file system interface is based on DOS 2.0 UNIX-style file handlers rather than the user-maintained file control blocks required by earlier versions of DOS. As a result, files may be referenced by their path names and the current directory changed at will.

PC/Forth is organized within a 64K address space, although another LMI package, PC/Forth+, is available that uses a 32-bit stack and can address all of memory for both data and operations. PC/Forth and PC/Forth+ use the same command structure and can compile each others' source code, though the different stack width may require you to make some minor modifications in the program.

In PC/Forth, *long* operators can read and write data to high memory, either directly to and from the disk or to and from low memory. The *long* operators use segment addressing, with Forth commands available to retrieve segment addresses. (Separate segments are also used for the stacks.) To optimize memory usage, system utilities are stored in precompiled form as binary overlays. Commands are provided so that users can also easily create relocatable binary overlays for their own program modules. These modules can be accumulated as a library of run-time routines that can be called as needed.

The screen editor is fast and complete, with control key commands in the WordStar dialect. The screen editor requires a fair amount of memory, which is why it is a binary overlay. For program development, a supplemental small editor would be useful.

The PC/Forth documentation is the best packaged of the Forth systems reviewed. Published in a sturdy IBM PC-style binder and box, the 383-page *PC/Forth Language Reference Manual* has the usual chapters describing the source editor and assembler. In addition, the manual contains a very complete 259-page description (glossary) of all Forth words needed by an applications programmer; there is also a more limited glossary of words of interest to the system programmer. All words are listed in the index. Further chapters describe the use of binary overlays, how to create turnkey applications, and how to use the MS-DOS interface.

A variety of add-ons are available for PC/Forth, including a very strong graphics extension, a debugger, a native code compiler that compiles Forth source code down to machine code, and a metacompiler that allows users to restructure the system according to specific needs and strip headers from application packages to gain more room and more program security.

Laboratory Microsystems requires no royalties or license fees for program products written in PC/Forth, provided that the end user has no access to the underlying Forth. LMI does request that developers submit a copy of the product and get explicit permission before distributing programs written in PC/Forth.

Laboratory Microsystems maintains an on-line bulletin board during nonbusiness hours on which users can leave requests for technical assistance. Answers are normally posted within 24 hours. In addition, telephone support is available during morning business hours.

MicroMotion MasterForth

MasterForth (v. 1.0) is a high-quality Forth software development system for the IBM PC developed by long-term Forth vendor MicroMotion and designed to be entirely compatible with the Forth-83 Standard. MasterForth is well integrated into the DOS environment. The program is stored as a DOS COM file and allows access to files by way of DOS 2.0-style file handlers. Files may be referenced by their full directory path names and the current directory changed at will.

Certain deficiencies exist in the DOS interface. To be opened or created, for example, a file must have an extension (the up-to-three-character portion of a file name following a period. No direct accommodation is made for interpretation of text files, though the *STREAM* package allows reading text files and the *EVAL* function permits interpretation of a string so that a user could write a word to read source code from text files. System source is distributed in the form of files of Forth blocks.

To optimize memory consumption in a system designed for small-model 64K addressing, MasterForth maintains

Software Development Tools

If:

Manufacturer-Compatible, Fully-Supported Cost-Effective, Cross and Native Development tools are important to you!

Then:

Let us tell you about MICROTEC RESEARCH's extensive line of field-proven software development tools for serious software developers.

C
Pascal
Assemblers
Simulators

Manufacturer Compatible

MICROTEC RESEARCH has been providing flexible and economical solutions for software developers since 1974. We've grown with the industry as our cross software development tools have transformed many large and small computers into powerful cross development systems for microprocessor applications. Our software tools are manufacturer compatible. Therefore, software made using manufacturers development systems may be more productively used in the MICROTEC RESEARCH cross-development environment.

Fully-Supported

MICROTEC RESEARCH's products are continually maintained and updated based upon the experience of thousands of installations worldwide. Revisions and modifications are distributed to licensees in warranty or covered by a service contract. Upgrades, which are major enhancements to a product's capabilities, are available at a significant discount. Our update/upgrade policy assures users they'll always be current with the fast moving world of microprocessor development.

Effective Differences

Beginning with product concept, through development, quality assurance, and post-sales support — **Quality, Compatibility,** and **Service** are the differences which set MICROTEC RESEARCH apart from the others.

Start Saving Time & Money

If you are a serious software developer, shopping for software development tools, call or write today for more information. **800-551-5554** In CA call **(408) 733-2919**

Target Microprocessors
- 8086/80186
- 8096
- 8080/8085
- 8051
- 8048
- Z8002
- Z80
- Z8
- NSC 800
- 9900...
- 68000/08/10
- 6809
- 6800/01/02
- 6805
- 6301
- 6305
- 6500
- G65SCXX, G65SC1XX
- R65C00, R6500/1...
- others

Host Computers
- DEC VAX, PDP-11
- DG MV-Series
- DG Eclipse
- Apollo
- UNIX Systems
- IBM PC
- Data General/ONE
- HP 150
- DEC Rainbow
- others

Software Tools
- C Compilers
- Pascal Compilers
- Assemblers
- Linking Loaders
- Librarians
- Conversion/Download
- Communications
- VT-100 Emulator

for Serious Software Developers

MICROTEC™
RESEARCH

3930 Freedom Circle, Suite 101, Santa Clara, CA 95054
Mailing Address: P.O. Box 60337, Sunnyvale, CA 94088
(408) 733-2919 • Telex (ITT) 4990808

MICROTEC is a trademark of Microtec Research, Inc. Santa Clara, CA

CIRCLE 75 ON READER SERVICE CARD

General information

Manufacturer and product	Forth standard	Editor	Assembler	Debugger	Kernel source	Manual pages	Disks	Multitasking	Access to extended memory	Access to DOS files	Interpret text files	Redirection works	Double integer	Triple or quad integer	Single floating point	Double floating point	Hardware floating point
IBM PC																	
Forth Inc. polyFORTH II IBM	79[1]	yes	yes	no	0[2]	960	2[3]	yes	no	yes[13]	no	no	yes	no	no	no	yes
polyFORTH II MSD	79[1]	yes	yes	no	0[2]	906	2[3]	yes	no	yes[14]	no	no	yes	no	no	no	yes
Harvard Softworks HS/FORTH 2.05	83	yes	yes	yes	no	299[9]	1	yes	yes[19,21]	yes	yes	yes[16]	yes	no	no	no	yes
Laboratory Microsystems PC/Forth 3.0	83	yes	yes	no	no	380	1	yes	yes[19,22]	yes	yes	no	yes	no	no	no	no
MicroMotion MasterForth 1.0	83	yes	yes	yes	no	325	1	no	yes[19,22]	yes	yes	yes[16]	yes	no	0	0	no
Mountain View Press MVP-FORTH 1[5]	79	yes	yes	yes	yes[2]	204	8	no	yes[19,22]	yes	no	no	yes	no	no	no	yes
Next Generation Systems NGS Forth 2.1.1	79[7]	yes	yes	yes	no	208	2	no	yes[19,21]	yes	yes	no	yes	no	no	no	no
No Visible Support Software F83 2.1.0	83	yes	yes	yes	yes	10	1[4]	yes	no	yes	no	no	yes	no	no	no	no
Shaw Laboratories TaskFORTH 8088 v. 1.00A	83,79	yes	no[6]	no[6]	no	—[9]	1[15]	yes	yes[21]	yes	no	no	yes	yes	no	no	no
Sunset Technology ST-FORTH 1.3	83	yes	yes	no	yes[2,10]	148	1	no	yes[20,21]	yes	no	no	yes	no	no	no	no
Ubiquitous Systems u4th 1.0	83[11]	yes	no	no	0[12]	429	1	no	no	yes	yes	yes[17]	no[18]	no[18]	no	no	no
Unified Software Systems UNIFORTH	83	yes	yes	yes	0[2]	500[9]	3	yes	yes[21]	yes	yes	no	yes	no	yes	yes	yes
Macintosh																	
Creative Solutions MacForth 2.3	79	yes	yes	yes	no	509	2	no	yes	yes	yes	N/A	yes	yes	no	yes	yes
MicroMotion MasterForth 1.0	83	yes	yes	yes	no	325	1	no	yes	yes	yes	N/A	yes	no	0	0	N/A

0 = Option.
N/A = .Not applicable.

1. polyFORTH is also *Starting FORTH* compatible.
2. Source is written in Forth and Forth assembler.
3. One of the two disks with polyFORTH is on-line documentation which corresponds with the source.
4. F83 is distributed with all source present in compressed files. Uncompressed, they take two disks.
5. Includes the additional MVP-FORTH PADS and the PADS options.
6. Available in advanced package.
7. Also includes a fig-Forth compatibility option.
8. Includes a print spooler utility, but not generalized multitasking.
9. We haven't seen part or all of the manuals for these systems.
10. Can look at but not compile.
11. u4th is somewhat compatible with the Forth-83 Standard, but with a number of variances from it.
12. Source to u4th is written in the C language.
13. A Forth utility interfaces directly to the DOS file system, allowing files to be accessed.
14. A single file named FORTH.SCR is opened and allowed access as Forth blocks. The new version has an extended DOS handling function.
15. Two for the advanced package.
16. Handles I/O redirection from the DOS command level but not from within Forth.
17. u4th allows I/O redirection both at the DOS command level and from within Forth.
18. Double and higher integers are available only on CPUs where the natural word size matches––even then access uses 16-bit Forth operators (*, etc.), a non-Standard feature.

Codes for accessing extended memory:

19. Long character fetch and store operators such as LC@ and LC! (names and functions differ).
20. Access using a RAM disk and the Forth block buffers.
21. Access by way of a segmented memory model organization.
22. Uses binary overlays for code (doesn't increase access to memory but improves memory usage).

Table 1.

utilities, such as the editor, assembler, debugger, and beheader, as binary overlays. Provision is made for the user to create custom overlays. Two additional memory optimization mechanisms are available: declaring certain code to be transient (allowing it to be discarded without affecting resident code compiled later) and declaring the headers of words transient while the code stays resident (allowing word headers to be discarded after all references have been compiled).

MasterForth is also well optimized for execution speed. It is organized internally according to a structuring scheme known as direct-threaded code. Most Forths today use indirect-threaded code or even token-threaded code, which have their own advantages (typically requiring less space) and disadvantages (slightly slower execution speed).

Developers should note that a one-time license fee of $1,000 is required for each program product, even if the program is distributed at no cost.

MasterForth is accompanied by the 110-page *MasterForth Reference Manual* and by the 224-page textbook *Mastering FORTH* by Anita Anderson and Martin Tracy. *Mastering FORTH* is a well-written introduction to the Forth-83 Standard, and the book describes itself as exactly matching the MasterForth system. The reference manual includes details particular to MasterForth: file interface, utilities, system internals, and a description (glossary) of system words.

Miller Microcomputer MMSFORTH

Since this company was in between versions at the time of this review, the revised version of MMSFORTH was not available for review. What follows is a brief description of the new version.

Miller Microcomputer Service's MMSFORTH (v. 2.4), like polyFORTH, takes over the machine completely and is incompatible with DOS and DOS-based files. Having complete control offers some advantages in terms of diskette capacity and video control but brings up the compatibility problems previously discussed. For example, if you are using a nonstandard hard disk or equivalent, such as the Bernoulli Box, you would have to write your own driver. Iomega provides a driver with the Bernoulli Box, but it (of course) is written as a DOS-based program.

Within these limitations, MMSFORTH is a responsive language. It is not a Forth-83 Standard, but it adheres generally to the Forth-79 Standard. The package has a long history, tracing back to TRS-80 model 1. Indeed, MMS continues to support MMSFORTH on TRS-80 models 1, 3, and 4; the MMS TRS-80 Forth is, within hardware limitations, the same as the IBM MMSFORTH reviewed here. The continued development of MMSFORTH has brought it to a high level of polish. The tie-on packages include not only Forth utility files but also support software such as a word processor, a data base system, and a telecommunications package. Thus the restriction to what amounts to the MMS operating system is not so severe as it might otherwise be.

The documentation is comprehensive, with all commands defined in an alphabetic list; command names are also grouped by function, and the commands a beginner should master first are identified in boldface. The manual is in a large, three-ring binder and is indexed.

The package includes an assembler and a breakpoint routine. Source is provided for extensions and the upper half of Forth but not for the innermost kernel. The program is restricted to 64K, but words to use upper memory for data storage are provided.

Double-precision integers come with the basic package. Arbitrary precision integers (you specify the number of bytes) and floating points (with 8087 support) are included on optional utility disks. On IBM computers, floating point operators can utilize the ROM routines.

Bootable and precompiled applications are easily prepared via a *CUSTOMIZE* word. Binary overlays are not supported directly but are examples in source code on the utility disk.

Developers should note that a corporate site license is required for producing pro-

Lattice® Products Are Your Best Investment

Lattice TopView Toolbasket—Provides more than 70 functions that eliminate the need for extensive use of assembly language when interfacing with TopView. Includes functions to control window, cursor, pointer, and printer operations. Also provides access to cut-and-paste facility and debugging services.	$250.00
LMK™—Automated product generation utility for MS-DOS, similar to UNIX "Make"	$195.00
C-Sprite™—Symbolic debugger for programs written in Lattice C or assembly language	$175.00
C Compilers—MS-DOS and PC-DOS—The industry standard	$500.00
Panel—High performance screen design and screen management system. Available for MS-DOS, PC-DOS, and UNIX systems	$295.00
dBC II™ or dBC III™—C function libraries supporting indexed files that are compatible with dBASE II® or dBASE III™	$250.00
C-Food Smorgasbord™—Includes a BCD decimal arithmetic package, I/O functions, IBM-PC BIOS interface, terminal independence package, plus many other utility functions	$150.00
Text Management Utilities—(GREP/DIFF/ED/WC/EXTRACT/BUILD)—Essential programming and documentation aids in any environment	$120.00

With Lattice published products you get "Lattice Service" including telephone support, free updates during the warranty period, notice of new products and enhancements when you register, and a money-back guarantee. Corporate site license agreements are available.

Call us today for further information on these, or our many other C programs, utilities, and tools!

Lattice, Inc.
P.O. Box 3072
Glen Ellyn, IL 60138
Phone (312) 858-7950
TWX 910-291-2190

International Sales Offices

Belgium: Softshop. Phone: (32) 53-664875
England: Roundhill. Phone: (0672) 54675
Japan: Lifeboat Inc. Phone: (03) 293-4711

CIRCLE 16 ON READER SERVICE CARD

grams from MMSFORTH, along with a license fee of $500 for 50 units or a one-time fee of $5,000 for an unlimited number of units. The fee applies whether programs are sold or distributed at no cost and is for distributing a run-time version of MMSFORTH (the end user having no access to the underlying Forth, only to the application program).

The normal price is for a one-person, one-computer license; a corporate site license is an additional $1,000 for unlimited users and copies within a single building. Additional buildings are $500 each.

Mountain View Press MVP-FORTH

Mountain View Press sells a wide variety of Forth systems, documents, and materials; some of these are from other publishers, some are published by Mountain View Press. The house-brand Forth is known as MVP-FORTH. Always a capable, well-documented Forth system, it now calls itself a virtual system (MVP-FORTH's terminology for binary overlays), allowing certain utilities to be read, already in precompiled form, from disk into the system's 64K addressable memory.

MVP-FORTH (v. 1.3) is not an Forth-83 Standard, though it is compatible with the Forth-79 Standard. Mountain View Press states that its Forth kernel will be unchanged in the future.

MVP-FORTH is available with two optional system components known collectively as the MVP-FORTH Professional Applications Development System (PADS). The first component of PADS is a package of Forth programming aids. The other optional PADS component is a unit made up of the metacompiler and virtual loader systems.

The Forth programming aids are a set of routines designed to speed program development and extract subroutines or an entire application program for cross compilation. Included in this optional package are a translator, which provides one-to-one translation (a form of decompilation, not recompilable) of high-level Forth words in memory; a callfinder, designed to find all calls to a specified word or set of words; a decompiler, capable of decompiling back to compilable source on disk; and a subroutine decompiler, which may, for example, extract a working subroutine in source form to disk, together with all its variables, constants, and called subroutines.

The metacompiler system supports a range of operations varying from recompiling the MVP-FORTH kernel after modifying some of the words to creating an entirely different system. The virtual loader system provides a technique for precompiling subsystems (overlays) in upper memory and storing them on disk until needed.

MVP-FORTH is at an intermediate stage of integration into the DOS environment. The system disk contains a DOS file system, with the Forth system appearing therein as a COM file. However, the remaining diskettes supplied with the MVP-FORTH system contain no file system but are organized simply as Forth blocks. Several of these diskettes contain a stand-alone boot image of MVP-FORTH.

The MVP-FORTH system is reasonably well documented, with a 56-page *MVP-FORTH Virtual System* manual, a 55-page *Forth Programming Aids* manual, and a 23-page *PADS Metacompiler and Virtual Loader Systems*. Of the PADS options, the *Floating Point System* manual is 28 pages long, the *Graphics Manual* is 13 pages, and the *MS-DOS File Interface Supplement* is 29 pages. Also included is a system source listing. In addition, Mountain View Press has published several books on MVP-FORTH, including a detailed reference on the MVP-FORTH system. MVP-FORTH, PADS, and the PADS options are distributed on a total of eight single-sided, double-density diskettes.

With the exceptions of the metacompiler system and the programming aids, the MVP software is in the public domain. Thus no royalties or license fees are required for products produced in this language.

Next Generation Systems NGS Forth

Next Generation Systems' NGS Forth, like some of the other Forths, utilizes seg-

Mystic Pascal

Fastest Compiler on Earth—$39.95!

Mystic Pascal compiles at 100,000 to over 1,000,000 lines per minute! How? It takes a short cut called *incremental compilation.* Compared to earlier Pascals, the effective speed is astronomical. Give the Compile command for a 1000 line program, and you probably can't lift your finger from the keyboard before the compiler flashes—DONE!

640K of storage not 64K. Are you fed up with being forced to shoehorn your programs into 64K? You won't need mystical powers to run your program in the full 640K that we allow—code, data and stack.

Interactive Pascal. You can enter Pascal statements directly and see the results instantly. It works like a Basic interpreter but it's a true compiler!

Optimized 8086 Code. Mystic Pascal produces true 8086 object code. The TWO code optimizers run in the background so they don't slow you down. Thanks to another breakthrough, our software floating point arithmetic runs faster than Intel says is possible on the 8088/8086 chips.

Real Multi-Tasking Pascal. Advanced programmers may write truly concurrent Pascal programs by simply starting Pascal procedures. Up to 100 concurrent procedures can communicate by passing messages through queues.

ISO Standard Pascal. Educators in particular need a truly Standard Pascal for their students. And learning is made easier by the Help Windows which describe Pascal.

Mystic Canyon Software
P.O. Box 1010
Pecos, New Mexico 87552

Place your order by phone today—(505) 988-4214 or mail the coupon.
Requires an IBM Personal Computer or true compatible with 256K. Not copy protected.

Rush me the Mystic Pascal System with diskette and manual!

Name ______

Address ______

City ______ State ______ Zip ______

☐ Check/Money Order ☐ Visa ☐ Mastercard

Price is $39.95 plus $4 shipping. CODs and Purchase Orders are NOT accepted. Outside US & Canada shipping charge is $20. Payment must be in US funds on a US bank.

Card ______ Exp. ______

Signature ______

CIRCLE 46 ON READER SERVICE CARD

ment addressing to make full use of the IBM memory space, even though the program is restricted to 64K.

NGS Forth places the stack in a separate segment and allows the extra segment to be set to point to data elements in extended memory. This makes better use of the 8088's 1MB addressing space without incurring the overhead of full large-model organization of the system.

NGS Forth provides a good degree of DOS integration although at the DOS 1.0 file control block level. NGS Forth is capable of accessing both Forth blocks and DOS text files and can convert from one to the other. NGS Forth is also capable of interpreting text files directly. It does not support DOS 2.x path names.

NGS Forth is not Forth-83 Standard. Normally it is Forth-79 Standard, but it also has the option of switching to a Forth Interest Group look-alike mode for a measure of compatibility with fig-Forth, an early standardized Forth.

NGS Forth's 208-page manual is published in an IBM PC-size binder. It contains sections describing the operation and structure of NGS Forth, a glossary of system words, and an index. Appendices describe specific tools, such as line and screen editors, an assembler, and debugging aids. The system supplied for review was v. 2.1.1, while the accompanying manual was for v. 1.1; we are unable to say what system features were added or changed for v. 2.

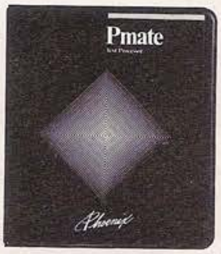
Next Generation Systems requires no royalties or license fees for the distribution of programs written in NGS Forth provided that the user has no access to the underlying Forth.

No Visible Support Software F83

Those who use and appreciate Forth must give much credit for its early growth to those public-spirited founding members of the Forth Interest Group who in the late 1970s organized implementation teams to bring up their newly defined fig-Forth on major microprocessors. The fig-Forth systems were then placed in the public domain. The availability of inexpensive Forth systems where there had been none made it possible for people to learn about and use Forth. fig-Forth defined its own standard and may be considered the first standard Forth to achieve widespread use.

Many fig-Forth systems are still in use. However, the Forth-79 Standard defined a Forth of greater power and capability than fig-Forth. A further blow to fig-Forth was struck by the drafting of the Forth-83 Standard.

To blaze a trail into the Forth-83 Standard world and stimulate commercial vendors to adopt the new standard, Forth collaborators Henry Laxen and Michael Perry revived the ideals of fig-Forth to create the Forth-83 Model Implementation, commonly known as F83. F83 systems are available for 8080, 8086, and



One Text Processor Beats Pmate™. The New Pmate.

No other PC/MS-DOS® text processor lets you edit files at the same time that you're compiling a program. Or, call up your editor while inside another application. Or, put long text processing jobs in the background and run other programs while they execute.

No other text processor offers you built-in macros that give you C and FORTRAN language-specific editing features. Brace matching. Movement, deletion, and copy by function. Keyword insertion. Error processing, and many more. Plus, since you get macro source code for all language-specific features, you can modify or add features as you wish.

And, no other text processor lets you use 1-2-3®-like menus, mouse pop-up menus, command-driven operation, or any combination of the above. With Pmate's 170-command macro language, you can program a single key to handle multiple command sequences. Call up other macros or common statements. Pass arguments to macros. Or, set up your own personalized text processing system with all your favorite features.

Plus, you get full DOS 2.x pathname support. 100 numeric variables. A number stack for storing intermediate results. Arithmetic and logical operations. Structured control statements. As well as numerous string, text and document manipulation functions.

And, of course, you get standard Pmate features such as full-screen single-key editing. Automatic disk buffering. Ten auxiliary buffers. Horizontal and vertical scrolling. A "garbage stack" buffer for retrieval of deleted text.

Pmate is priced at $225 and is available for 8086-based micros running MS-DOS. A custom version is available for the IBM® PC, TI Professional,™ Wang Professional,™ and DEC Rainbow.™

Find out more about the new Pmate today! Call (800) 344-7200. In Massachusetts (617) 762-5030. Or, write.

Phoenix Computer Products Corp., 1420 Providence Highway, Suite 115, Norwood, MA 02062

PROGRAMMERS' PFANTASIES™ BY
Phoenix

Pmate and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation. MS-DOS is a trademark of Microsoft Corporation. 1-2-3 is a registered trademark of Lotus Development Corporation. IBM is a registered trademark of International Business Machines. TI Professional is a trademark of Texas Instruments Incorporated. Wang Professional is a trademark of Wang Laboratories, Incorporated. DEC Rainbow is a trademark of Digital Research Corporation.

CIRCLE 18 ON READER SERVICE CARD

68000 processors running CP/M and 8088 processors running MS-DOS. The version reviewed here is 2.1.0.

Like fig-Forth, F83 is in the public domain. Laxen and Perry have published the following policy statement: "In order to promote widespread distribution and use of this model, we have placed it in the public domain. We encourage its use, reproduction, extension, and improvement, especially the latter two."

F83 is a Forth-83 Standard and is supplied with all system source, including a metacompiler. The system is multitasking and comes with various utilities: an assembler, a *Starting FORTH*- compatible editor, a decompiler, and a source-code locator facility.

As implied by the vendor name, F83 has no vendor support and is supplied with no documentation other than the system source itself and a couple of text files containing hints. As Laxen and Perry state, "All [systems] are available as is, with no support, handholding, questions answered, warranties, guarantees, assurances, refunds, or any recourse whatsoever."

F83 is moderately integrated into the DOS environment, providing operators to open and access DOS files at the DOS 1.0 file control block level and storing its source in files of Forth blocks. F83 includes no facilities for interpretation of text files and does not support DOS 2.x path names.

The MS-DOS version of F83 is distributed on a single double-sided, double-density diskette containing a number of Huffman-encoded files. When unpacked, these files expand to fill two additional floppies, a procedure requiring approximately one hour.

Benchmark results (sec)

Manufacturer and product	Loop	Subtraction test	Multiply test	Divide test	Move	Compare test	Sieve 1	Sieve 2	Sieve 3	Sieve 4	Sieve 5
IBM PC											
Forth Inc. polyFORTH II IBM	1.118	3.641	4.473	4.723	3.679	4.456	54.898	36.907	66.026	66.384	74.070
polyFORTH II MSD	1.063	3.470	4.250	4.463	3.471	4.235	52.502	35.214	62.855	63.195	70.510
Harvard Softworks HS/FORTH 2.05	1.204	3.284	4.134	4.594	3.636	4.131	47.818	33.841	53.530	56.934	61.522
Laboratory Microsystems PC/Forth 3.0	0.779	3.400	4.265	4.621	3.223	3.991	52.209	33.419	62.078	64.608	71.573
MicroMotion MasterForth 1.0	0.638	2.691	3.573	9.584	3.305	3.214	45.028	29.570	53.583	54.520	60.809
Mountain View Press MVP-FORTH 1	1.098	3.570	4.461	4.589	3.801	4.214	55.010	—[1]	64.680	65.709	72.671
Next Generation Systems NGSForth 2.1.1	0.992	3.471	4.389	4.620	3.541	4.041	53.485	35.383	64.783	63.984	71.529
No Visible Support Software F83 2.1.0	1.172	4.462	7.619	37.367	4.463	5.206	67.796	43.923	79.863	80.430	89.008
Shaw Laboratories TaskFORTH 8088 v. 1.00A	0.893	3.553	4.338	5.691	3.471	4.233	55.147	35.202	64.597	65.986	73.130
Sunset Technology ST-FORTH 1.3	0.850	3.470	4.303	5.130	3.471	4.049	54.754	34.498	64.808	64.511	71.953
Ubiquitous Systems u4th 1.0	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]	—[2]
Unified Software Systems UNIFORTH	0.779	3.426	4.281	4.462	3.371	4.061	54.145	34.128	63.807	65.141	71.826
Macintosh											
Creative Solutions MacForth 2.3	0.700	2.100	5.100	N/A	2.900	2.500	34.400	21.300	41.400	45.900	50.900
MicroMotion MasterForth 1.0	0.400	1.400	1.700	N/A	1.900	1.900	25.200	16.800	29.700	30.200	32.900

N/A = Not applicable.

[1] Running Sieve 2 crashes MVP-FORTH.

[2] Lacking a XENIX system for the IBM PC, u4th could not be tested or benchmarked.

Note: The benchmark routines used in this review are essentially identical with those used in a recent article in *BYTE* by Ernie Tello ("Software Review: polyFORTH and PC/FORTH," November 1984, pp. 303-314). The loop, subtract, multiply, divide, move, and compare tests originally appeared in *FORTH Dimensions* (issue III/1, p. 11). The various Eratosthenes Sieve programs are derivatives of the Sieve of Eratosthenes algorithm appearing in the September 1981 *BYTE,* which Tello selected for his review. Sieve 1 is the basic Sieve of Eratosthenes algorithm. Sieve 2 is an improved Eratosthenes Sieve by Don Colburn. Sieve 3 is an Eratosthenes Sieve described as "optimized." Sieve 4 is a Sieve which makes use of a *CREATE DOES>* character array. Sieve 5 is a Sieve described as using an "optimized array." Timings on the MS-DOS systems were performed by combining a count maintained by the 55-millisec timer interrupt with low-order bits read directly from the IBM PC's 8253 chip, reset to mode 2. Although up to microsecond resolution is theoretically available using this technique, in practice resolution to about a millisecond was attained. Stand-alone polyFORTH resets the 8253 to interrupt every millisecond; in this case, polyFORTH's built-in timer utility—also estimated to have approximately millisecond resolution—was used.

Table 2.

ORDER COMPUTER LANGUAGE BACK ISSUES — WHILE THEY LAST!

COMPUTER LANGUAGE
THE FOUNDERS OF BASIC UNVEIL THEIR NEW STRUCTURED LANGUAGE

PREMIER

The biggest collector's issue

- Basic Becomes a Structured Language—by Kemeny & Kurtz
- Programming in the UNIX Environment
- COBOL: Pride and Prejudice
- Exploring Ada and Modula-2

Exotic Language: SNOBOL

Interview: Charles Moore

COMPUTER LANGUAGE
PORTABILITY IN C

OCTOBER '84

- An Implementation Demonstrating C Portability
- The Evolution of ZCPR-Part I
- BATCH—A Powerful IBM "Language"

Exotic Language: PILOT

Interview: Donald Knuth

COMPUTER LANGUAGE
ENHANCING SOURCE CODE CONTROL UNDER UNIX

NOVEMBER '84

- Enhancing Source Code Control under UNIX, Part I
- Natural Language Processing and LISP
- Building Portable Programs

Exotic Language: OCCAM

Interview: Gary Kildall

COMPUTER LANGUAGE
EXPLORATORY PROGRAMMING

DECEMBER '84

- Exploratory Programming
- Fred: A Language within Framework
- Six Pascal Compilers Compared

Exotic Language: OMNI

Interview: Bill Godbout

COMPUTER LANGUAGE
MACROS AND PROCEDURES

JANUARY '85

- Macros and Procedures
- Extensibility in Forth
- The IIIrd Dimension-Programming in dBASE III

Exotic Language: Transaction Application Language

Interview: Sol Libes

COMPUTER LANGUAGE
SPECIAL C ISSUE

FEBRUARY '85

C Language Special Issue

- Twenty-One C Compilers Compared
- The Standardization of C
- C to Assembly Interface

Exotic Language: MUMPS

Interview: P.J. Plauger

COMPUTER LANGUAGE
MODIFYING MS-DOS DEVICE DRIVERS

MARCH '85

- Modifying MS-DOS Device Drivers
- Hashing out FORTH with Charles Moore
- Programming Macros in C (Debugging)
- Sol Libes Covers Multiprocessing Systems

COMPUTER LANGUAGE
RECURSIVE PROCEDURES

APRIL '85

- Recursive Procedures
- Sorting by Dispersion
- COBOL Compiler Analysis
- C on the Macintosh

Exotic Language: APT

Interview: Gordon French

COMPUTER LANGUAGE
NIKLAUS WIRTH AND DONALD KNUTH

MAY '85

Exclusive Interview: Niklaus Wirth and Donald Knuth

- Twenty-nine BASIC Interpreters and Compilers Compared
- Macros In BASIC

Exotic Language: Clascal

COMPUTER LANGUAGE
RECURSING FORTH

JUNE '85

- Forth: Twithe Curthed, too!
- Porting the UNIX Utilities
- Symphony Command Language

Exotic Language: COMAL

Interview: Ward Christensen

Only a limited quantity of magazines is available, so order today. To receive your back issues, just fill out this coupon and mail it back with a check for $5.00 per issue.

Foreign orders: Add $3.00 for airmail.

Premier ______ copies × $5.00 = $ ______
Oct. '84 ______ copies × $5.00 = $ ______
Nov. '84 ______ copies × $5.00 = $ ______
Dec. '84 ______ copies × $5.00 = $ ______
Jan. '85 ______ copies × $5.00 = $ ______
Feb. '85 ______ copies × $5.00 = $ ______
Mar. '85 ______ copies × $5.00 = $ ______
Apr. '85 ______ copies × $5.00 = $ ______
May '85 ______ copies × $5.00 = $ ______
Jun. '85 ______ copies × $5.00 = $ ______
July '85 ______ copies × $5.00 = $ ______
Aug. '85 ______ copies × $5.00 = $ ______
TOTAL $ ______

NAME ______
COMPANY ______
ADDRESS ______
CITY, STATE, ZIP ______

Send payment and coupon to:

8

COMPUTER LANGUAGE
Back Issues
131 Townsend St.
San Francisco, CA 94107

BRING BACK
ERASED FILES

FILE SAVIOR ™

is a disk utility you can't afford to be without!

- Makes recovering an erased file easy... as long as you haven't written a new file on top of it
- Display and edit any disk sector – in hex, Ascii or Ebcdic
- Search a file or your whole disk for a text string
- IBM PC/XT/AT and true compatibles
- DOS 1.0 thru 3.1
- Any DOS-formatted diskette or hard disk – up to 33 meg

49⁹⁵ VA residents add 4% sales tax

Technisoft

1710 Allied Street, Suite 37
Charlottesville, VA 22901
(804) 979-6464

FAST SCREEN
OUTPUT FOR
TURBO PASCAL

FASTSCREEN ™

is a set of inline assembler and Pascal procedures for Turbo Pascal users.

- Display an entire screen or window almost instantly
- Process multi-field input screens that give your user full cursor control
- Sample program uses Conway's LIFE to illustrate use of procedures
- All source code included
- Color and monochrome support
- IBM PC/XT/AT and true compatibles
- DOS operating system

29⁹⁵ VA residents add 4% sales tax

Technisoft

1710 Allied Street, Suite 37
Charlottesville, VA 22901
(804) 979-6464

Turbo Pascal is a registered trademark of Borland International

CIRCLE 83 ON READER SERVICE CARD

Shaw Laboratories TaskFORTH

George Shaw has long been a major contributor to the Forth community; he is a member of the Forth Standards Team and was one of the elected referees who completed the Forth-83 Standard. Shaw Laboratories Ltd. has now released a new version of its TaskFORTH system that conforms to the Forth-83 Standard as well as allowing Forth-79 Standard and *Starting FORTH* compatibility modes.

TaskFORTH (8088 v. 1.00A) is moderately integrated into the MS-DOS environment; it runs under DOS as an application program, provides the ability to access DOS files (at the DOS 1.0 file control block level), and stores its source in files of Forth blocks. However, no capability is provided for interpretation of DOS text files and DOS 2.0 path names are not supported. TaskFORTH is capable of accessing absolutely-addressed disks that lack a DOS file system.

On top of DOS files (or absolutely addressed disks), TaskFORTH provides the capabilities of its own hierarchical file system and data base manager, allowing further internal structuring of DOS files. The file system supports both fixed contiguous and dynamic allocation with random, sequential, and keyed file access. Additional file types, record structures, and access methods can be defined and integrated into the system. Records can be divided into and accessed by named fields.

TaskFORTH is multitasking and allows multiple users as well. When an image of the TaskFORTH system is saved into a file, every task is provided with a vector allowing it to execute its own programs from a cold start. TaskFORTH's multiprogrammer is an efficient, round-robin scheduler; any task can do anything a user can. The only limit on tasks is the amount of available memory.

Various utilities accompany the TaskFORTH system—a video editor, an assembler, a decompiler, a disassembler, performance monitoring tools, and an interactive high-level debugger capable of debugging multiple tasks. A range of single-, double-, and triple-size fixed-point arithmetic operations are supported and optimized for speed.

Documentation for the new version of TaskFORTH was unavailable at press time and could not be reviewed.

Sunset Technology ST-FORTH

ST-FORTH (v. 1.3) is compatible with the Forth-83 Standard and is at an intermediate stage of integration into the MS-DOS environment. The program is supplied as an MS-DOS COM file and has the ability to access DOS files by way of DOS 1.0-type file control blocks. Words are provided to allow access to sequential and randomly structured files. However, ST-FORTH considers its normal method of source storage to be non-DOS absolutely addressed floppy disks containing blocks, similar to the access of the stand-alone systems such as polyFORTH and MMS-FORTH.

The ST-FORTH system includes a video editor, assembler, high-level decompiler, source code locator, system memory map, and other useful tools. In an unusual feature, ST-FORTH is supplied with all source for the system (including the kernel), organized as a single 267K file of blocks. Also included is a RAM disk that allows access, via the disk buffers, to memory beyond the normally addressable 64K.

ST-FORTH is accompanied by the 156-page *ST-FORTH Users Reference Manual*, which describes the system components and includes a description (glossary) of all words in the system.

Ubiquitous Systems u4th

u4th (v. 1.0) is a Forth-83 Standard that runs on the IBM PC under the XENIX operating system. XENIX, a version of AT&T's UNIX, was developed by Microsoft and ported to the IBM PC by The Santa Cruz Operation. We were unable to obtain XENIX for the IBM PC by press time and so cannot provide benchmarks; however, we deemed u4th to be a well-developed product that deserves a report.

u4th is designed to be a highly portable product. Like many UNIX systems, it is written entirely in C. u4th reportedly runs on processors as various as Intel 80286 and Motorola 68000, with the only source changes being in a couple of statements. Another design objective was fastest execution speed consistent with portability. u4th documentation claims the system is slower only by a factor of approximately two than fast assembly language Forths. Internally, u4th uses direct-threaded code.

Highly integrated into the UNIX environment, u4th allows access to all types of UNIX files and provides fully recursive I/O redirection within its own process. Interpreting Forth source from a text file is performed simply by redirecting input to come from that file; similarly, output to files can be performed by redirecting the output. u4th allows commands to be automatically passed through to UNIX ("forking a shell") if the input can't be found in the Forth dictionary. UNIX-style filters can also be run without leaving the u4th environment. It's worth noting that many of these capabilities are available to programs operating in the MS-DOS environment.

Due to limitations in the XENIX C compiler, u4th currently is available only by using small-model 64K addressing. Upgrades to large model will be made available to current customers "when such support can be offered." Large-model u4th will run with 32-bit stacks, a

NEW FOR THE MACINTOSH

PRESENTING THE MEGAMAX C COMPILER

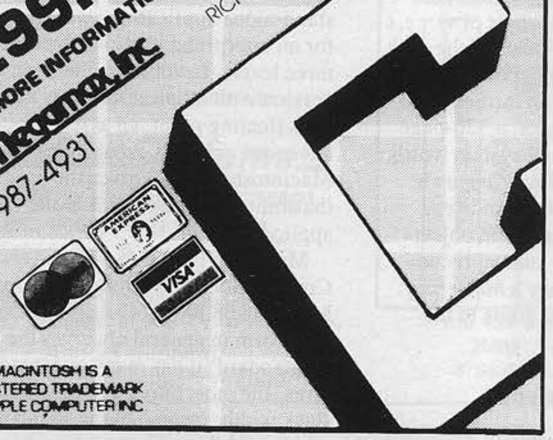
FEATURING:
• IN-LINE ASSEMBLY • ONE PASS COMPILATION • SUPPORT OF DYNAMIC OVERLAYS • FULL ACCESS OF MACINTOSH TOOLBOX ROUTINES • AND MUCH MORE...

DEVELOPMENT SYSTEM PACKAGE INCLUDES:
• FULL-SCALE IMPLEMENTATION (K&R) C COMPILER • THE STANDARD C LIBRARY • ROM ROUTINES LIBRARY • LINKER • LIBRARIAN AND DOCUMENTATION...

$299.95

DEALER AND USER GROUP INQUIRES INVITED

FOR MORE INFORMATION OR TO ORDER CALL OR WRITE:
Megamax, Inc
BOX 851521, DEPT. 1
RICHARDSON, TX 75085-1521
(214) 987-4931



MACINTOSH IS A REGISTERED TRADEMARK OF APPLE COMPUTER INC

CIRCLE 13 ON READER SERVICE CARD

BASIC
PASCAL

P-tral: BASIC to Pascal Translation Software

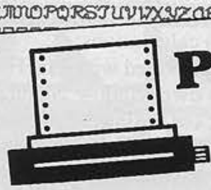
P-tral is the user-friendly translation software that converts Applesoft BASIC programs to Apple Pascal.

- **Translates any business, scientific, graphics and game software from source.**
- **Translation results approach 100%.**
- **Translates any sized program—Large or Small.**
- **Essential compile directives generated.**
- **PIONEER version only $125.00.**

REQUIREMENTS: 64/128K RAM, Dos 3.3 80-column card/Apple II, IIe, IIc Apple Pascal 1.1 or 1.2

To order or request more information contact:
WOODCHUCK INDUSTRIES®
340 West 17th Street, New York, NY 10011
(212) 924-0576 / (212) 206-6490

CIRCLE 55 ON READER SERVICE CARD



PrintShop
for your dot matrix printer

Letter Quality PLUS

PrintShop enhances your printing capabilities via resident drivers; providing letter quality printing on a dot matrix printer, multiple fonts, font editing, print to disk, monochrome graphics screen print, print **MacPaint** pictures, and much more. Runs under **PC-DOS** for **ImageWriter, Epson** and compatibles.

phone: 312/769-3757

$49.95

MasterCard VISA or prepaid

GKS

GKS Associates, Inc.
915 Lakeside Place · Chicago, Illinois 60640

CIRCLE 71 ON READER SERVICE CARD

WORKMAN & ASSOCIATES
USER-FRIENDLY SOFTWARE

JANUS ADA

R&R Software's Janus is the best form of Ada available for CP/M-80, CP/M-86, and PC/MS-DOS systems. Workman & Associates is the sole source of the CP/M-80 version of Janus. The full system comes with access libraries (for accessing system functions), 8087 support, transcendental functions, and more. Owners of this system can buy future updates at a discount. For MS-DOS orders, please specify DOS 1.1 or 2.0 compatibility. Price: MS-DOS $900, CP/M-80 $300; please inquire about our CP/M-86 version.

Zurich MODULA-2

This is a perfect learning compiler for those interested in MODULA-2. It's a true native code compiler for CP/M-80 that only lacks real numbers and conformant arrays to be full MODULA-2. Based on the ETH M4 compiler, it generates compact, efficient code. Price: only $160.00.

Workman & Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

Please request our new catalog. Workman and Associates ships over 200 formats from stock and we welcome COD orders. Please add $2.50 for shipping.

CIRCLE 68 ON READER SERVICE CARD

RUN/C:™ *Finally,* a C Interpreter

Available NOW for only **$149.95!**

Finally, a painless introduction to the C language. With **RUN/C: The C Interpreter** you can create and run C language programs in an environment as easy to use as BASIC.

RUN/C is C for the rest of us. It is a robust implementation of standard K&R. **RUN/C** is for both the beginner and professional.

RUN/C includes full floating point, 8087 support, structures, unions, casts and more than 100 built-in C functions.

With **RUN/C** you get all this with a command structure modeled after BASIC's using familiar terms such as EDIT, RUN, LIST, LOAD, SAVE, TRON, SYSTEM, etc.

Since **RUN/C** is a true interpreter it means that C programs can be written, tested and run within a single protected environment. It is a teaching tool and a source code debugger.

Here's more good news . . .

- Great documentation: a 400-page, easy-to-read manual filled with executable programs
- Array-index and pointer bounds checking
- Variable-trace and dump diagnostics PLUS an integral program profiler
- Full buffered and unbuffered file I/O
- Printer and asynch support
- Forking to your favorite full screen editor with automatic return to **RUN/C** with your edited program
- System Requirements: IBM® PC or compatible with PC-DOS 2.0 or MS™-DOS 2.0 or greater with ANSI.SYS.

Get things right the first time with **RUN/C: The C Interpreter.™**

For immediate delivery or more information, call:

1-800-847-7078
(in N.Y. **1-212-860-0300**)

or write: Lifeboat Associates™
1651 Third Avenue
New York, NY 10128

RUN/C is a trademark of Age of Reason Co.

CIRCLE 85 ON READER SERVICE CARD

variation from full compatibility with the Forth-83 Standard. Other differences from the Forth-83 Standard include definition of system words in lowercase, and only optional, high-level support for accessing files of Forth blocks.

The u4th product provides all system source, including the kernel, and makes use of UNIX's make program to control kernel recompilation. Since u4th stores its source in ordinary UNIX text files (which UNIX can manipulate through many well-designed utilities) and commands can easily be passed through to UNIX, u4th recommends using a UNIX text editor—such as the popular vi—for program development. For editing files of Forth blocks, u4th provides a video editor with a command set modeled after a subset of vi.

u4th documentation consists of the *u4th User's Manual*, consisting of 60 pages of "essential information," an introduction to u4th, and advanced topics; a 326-page description (glossary) of all system words (in the style of the *UNIX Programmer's Manual*); and 43 pages of appendices, including a 24-page appendix on object-oriented Forth. The u4th manual recommends Anderson and Tracy's *Mastering FORTH* as a supplemental guide to the Forth-83 Standard Forth.

Unified Software Systems UNIFORTH

Unified Software Systems is now completing conversion of its Professional Series UNIFORTH to the Forth-83 Standard. Because of this, we were unable to test the new Professional Series; instead, we ran our benchmarks on the UNIFORTH Sampler System, a public domain subset of the Professional.

A brochure describing the new UNIFORTH Professional System states that it includes both a line and video editor, an assembler, a decompiler, and a debugging trace facility. The Professional System is also a prioritized multitasking system. Software and hardware floating point support are standard, using IEEE format, and options such as a plotting package, EPROM programmer, and cross compiler (otherwise known as a metacompiler) are available. UNIFORTH does not support the creation of binary overlays or DOS 2.x path names.

Unified Software Systems requires no royalties or license fees for any program developed in the public domain sampler system or for any program developed in the Professional Series, provided the user has no access to the underlying Forth and the headers are stripped out using a word provided for that purpose.

Documentation for the new Professional Series UNIFORTH was unavailable at press time so we are unable to report on it. The UNIFORTH brochure states that all UNIFORTH systems come with a manual set designed to lead you painlessly from the basics to the heart of FORTH. The manual set is divided into the *User's Guide* and *Programmer's Guide* and totals more than 500 pages.

Professional Series UNIFORTH was scheduled for release this summer. UNIFORTHs for a number of other processors and operating systems are also reportedly available.

MACINTOSH FORTHS

Creative Solutions MacForth

MacForth was the second language available for the Macintosh, coming shortly after the release of MS-BASIC for the Mac. MacForth was the first language for the Mac that offered the ability to create stand-alone applications without the need for an interpreter. MacForth comes in three levels. Level 1 is an integer-only version without an assembler. Level 2 adds floating point, an assembler, and advanced graphics capabilities on the Macintosh. Level 3 gives the programmer the ability to create stand-alone applications.

MacForth is the Macintosh version of Creative Solutions' MultiForth for the Motorola 68000 processor. Although MacForth in general observes the Forth-83 Standard, it has many nonstandard features, the most important of which is the stack width: instead of the standard 16-bit width (which accommodates addresses up to 64K), the MacForth stack is 32 bits wide. Thus double-precision numbers use 64 bits instead of 32 bits. This provides a considerable range for fixed-point (integer-only) calculations.

The editor provided with MacForth operates in its own window. Standard Forth 1K blocks (16 lines of 64 characters each) are used to store source code; the mouse and pointer can be used with Macintosh cut-and-paste methods for very quick source code editing. If an error has occurred during compilation, the editor will present the block containing the error with the cursor at the exact spot where the error happened. Code compiled with the trace feature turned on can be debugged when executed. In this mode each word called will be displayed along with the stack contents at the time. Control-A can be used at any time to halt execution of a MacForth program.

Numerous extensions are provided with MacForth. With level 1, MacForth windows and menus can be created in the Macintosh environment, and lines, rectangles, and ovals can be drawn from the Macintosh toolbox routines. Level 2 adds region and polygon capabilities for creating and manipulating complex graphics. MacForth allows the coordinates of a window to be defined by the user.

The developer provides on-line support via a CompuServe special interest group. Access to this group is for registered users; the only fee is a surcharge on the

THE WORLD'S FASTEST MOST POWERFUL 8080 RELOCATING MACRO ASSEMBLER

SLRMAC™

ONLY $49.95

AT THIS PRICE, SHOULD YOU BE WASTING YOUR TIME USING SOMETHING ELSE?

This is what they said about Z80ASM, our Z80 assembler. Now the same features and performance are available in our Intel Mnemonic product. SLRMAC is compatible with M80 in .8080 mode, with many extensions. Too many features to list here.

To order or to find out more about our complete family of development tools, call or write:

SLR Systems

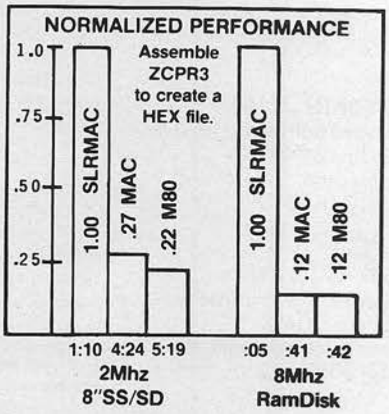
1622 N. Main St., Butler, PA 16001
(800) 833-3061, (412) 282-0864
Telex 559215 SLR SYS

C.O.D., Check or Money Order Accepted.

Add $3 shipping.
Z80 CP/M compatibility required.

". . a breath of fresh air . ."
Computer Language, Feb. 85

". . in two words, I'd say speed & flexibility",
Edward Joyce, Nov. 84
Microcomputing



CIRCLE 28 ON READER SERVICE CARD

INSIGHT™

EXPERT SYSTEMS

"INSIGHT is essentially the equivalent or better than any other tool available for the personal computer."

Paul Harmon, author of *Expert Systems, Artificial Intelligence in Business*

Turn your PC into an expert.

Give it Insight, or give it Insight 2. Both let you create knowledge base systems using any PC-compatible text editor.

Insight not only simplifies access to lots of information, it analyzes and offers solutions. For entry-level operators it's a perfect procedural training package to help build and implement knowledge base software.

Insight 2 is more than just an "expert." It's a knowledge base engineering tool with application capabilities. It can call up Pascal programs, read and write dBASE II® files, and its decision-making process can tie in directly to your existing databases. Run-only versions also can be developed and distributed.

Two unique packages from the same expert idea.

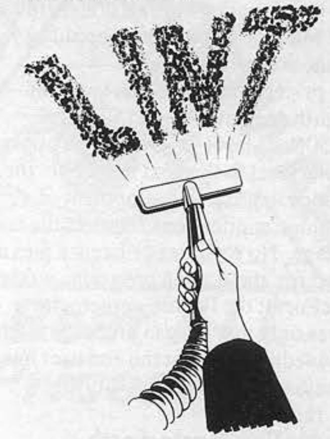
Level Five Research, Inc.

Insight™ ($95) and Insight 2™ ($485) run on the IBM® PC, DEC® Rainbow, and Victor® 9000.

4980 South A-1-A Melbourne Beach, Florida 32951 (305) 729-9046

CIRCLE 90 ON READER SERVICE CARD

REMOVE



from your **C** programs with

PC—LINT

PC—LINT analyzes your C programs (one or many modules) and uncovers glitches, bugs, quirks and inconsistencies. It will catch subtle errors before they catch you.

PC—LINT resembles the Lint that runs on the UNIX O.S. but with more features and greater sensitivity to the problems of the 8086 environment.

- Full K&R C
- Supports Multiple Modules—finds inconsistencies between declarations and use of functions and data across a set of modules comprising a program.
- Compares function arguments with the associated parameters and complains if there is a mismatch or too many or too few arguments.
- All warning and information messages may be turned on and off globally or locally (via command line and comments) so that messages can be tailored to your programming style.
- All command line information can be furnished indirectly via file(s) to automate testing.
- Use it to check existing programs, programs about to be exported or imported, as a preliminary to compilation, or *prior* to scaling up to a larger memory model.
- All one pass with an integrated pre-processor so it's very fast.
- Has numerous flags to support a wide variety of C's, memory models, and programming styles.
- **Introductory Price: $98.00 MC, VISA**
  (Includes shipping and handling) PA residents add 6% sales tax. Outside USA add $10.00.
- Runs on the IBM PC (or XT, AT or compatible) under DOS 2.0 and up, with a minimum of 128KB of memory. It will use all the memory available.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: IBM *(IBM Corp.)*, PC—LINT *(Gimpel Software)*, UNIX *(AT&T)*

normal CompuServe fee.

The benchmarks were all done with level 2 MacForth, but there should be no differences between levels.

The prices for the various levels of MacForth are: level 1— $150, level 2—$250, and level 3—$499. The upgrade from one level to the next costs only the difference in price. Development of stand-alone applications requires the level 3 package. No royalties or license fees are charged for application programs written in MacForth; the license agreement requires only that the use of MacForth be acknowledged and that the end user have no access to the underlying Forth.

MicroMotion MasterForth

MasterForth is a Forth-83 Standard for the Macintosh and is essentially identical to MicroMotion's MasterForth running on other computers. An assembler is provided with this package. The manuals for MasterForth consist of a copy of the book *Mastering Forth*, which is a generic guide for MicroMotion Forth products, and a small book that points out specifics of MasterForth for the Macintosh.

The editor provided with MasterForth operates from within the main window. Standard Forth 1K blocks are used for source code. MasterForth's editor does not use the Macintosh mouse; instead, control key sequences move the cursor and cut and paste lines of text. These sequences can be tedious to learn and awkward to use for programmers accustomed to the Mac's standard interface.

Macintosh toolbox routines are called by using traps. Examples are given, but doing such things as creating windows and menus with this system require access to Apple's huge *Inside Macintosh* document. The example code is sparse, so this system cannot be recommended for the novice. As an example of the level of expertise required, note that MasterForth addresses only 64K of memory. To address 512K, Macintosh's memory requires the use of the Mac's memory manager. Source code is provided for accessing the extra memory, but the programmer probably needs a good working knowledge of 68000 assembly language.

The stack used by MasterForth is the standard 16-bit width, with 32 bits for double precision arithmetic. As can be seen by the benchmarks, MasterForth is faster then MacForth, but the limitations of the MasterForth system should be taken into account. It should be noted also that Forth code is usually compact; more can be done in 64K than one accustomed to other languages would expect.

The ability to take snapshots of the MasterForth system and create stand-alone applications is provided in the package. To distribute copies of programs written with this system requires a $1,000 royalty fee for each program.

Forth products and manufacturers

PolyFORTH II—level 3—$600
level 4—$3,200
Forth Inc.
2309 Pacific Coast Hwy.
Hermosa Beach, Calif. 90254
(213) 372-8493

HS/FORTH—$250
Harvard Softworks
P.O. Box 69
Springboro, Ohio 45066
(513) 748-0390

PC/Forth —$100
Laboratory Microsystems
P.O. Box 10430
Marina del Rey, Calif. 90295
(213) 306-7412

MasterForth—$125—system
$60—relocator
$40—floating point
MicroMotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, Calif. 90025
(213) 821-4340

MMSFORTH—$179.95
Miller Microcomputer Services
61 Lake Shore Rd.
Natick, Mass. 01760
(617) 653-6136

MVP-FORTH—$500—PADS
$175—programmers kit
Mountain View Press
P.O. Box 4656
Mountain View, Calif. 94040
(415) 961-4103

NGS Forth—$72
Next Generation Systems
P.O. Box 2987
Santa Clara, Calif. 95055
(408) 241-5909

F83—$25
No Visible Support
2000 Center St.
Box 1344
Berkeley, Calif. 94704

TaskFORTH—starter—$250
advanced—$395
Shaw Laboratories Ltd.
P.O. Box 3471
Hayward, Calif. 94540-5953
(415) 276-5953

ST-FORTH—$35
Sunset Technology
1954 Menalto Ave.
Menlo Park, Calif. 94025
(415) 325-3680

u4th—$200
Ubiquitous Systems Inc.
13333 Bel-Red Rd. NE
Bellevue, Wash. 98005
(206) 641-8030

UniForth—$245
United Software Systems
P.O. Box 2644
New Carrollton, Md. 20784
(301) 552-9590

MacForth—level 1—$149
level 2—$249
level 3—$499
Creative Solutions
4701 Randolph Rd., Ste. 12
Rockville, Md. 20852
(301) 984-0262

MasterForth—$125—system
$60—relocator
$40—floating point
Micromotion
12077 Wilshire Blvd., Ste. 506
Los Angeles, Calif. 90025
(213) 821-4340

TURBO EDITASM

Introducing the first co-resident editor assembler for the IBM PC family. ***TURBO EDITASM (TASM)*** is significantly faster and easier to use than the IBM Macro-Assembler (MASM). Whether you are new to assembly language and want to quickly write a small assembly language routine, or are an experienced MASM user tired of waiting months to assemble large files, ***TURBO EDITASM*** will bring the excitement back to assembly language.

***TURBO EDITASM* IS MUCH FASTER:**

- How fast is ***TASM?*** The graph below shows relative assembly times for a 48K source file. For large files like this we blow MASM's doors off at 3 times their speed. For smaller 8K files we positively vaporize them at 6 times their speed.

TASM	(110 sec.)
MASM	(340 sec.)

- ***TURBO EDITASM*** is faster for the following reasons: **(1)** Written entirely in assembly language (unlike MASM). **(2)** Editor, assembler and source file always in memory so you can go instantly from editing to assembling and back. **(3)** Eliminates the time needed to LINK programs. Executable COM files can be created directly. (Also creates OBJ files compatible with the IBM linker).

***TURBO EDITASM* IS EASIER TO USE:**

TASM includes many other features to make your programming simpler.

- Listings are sent directly to screen or printer. Assemblies can be single stepped and examined without having to leave the editor.
- Access the built-in cross reference utility from the editor.
- Full support of 186 and 286 (real mode) instructions.
- Both Microsoft and 8087 floating point formats are supported. 8087 and 287 instructions supported directly without macros for faster assembly.
- Calculator mode: Do math in any radix even using symbols from the symbol table.
- Direct to memory assembly feature lets you test execute your code from editor.
- Coming soon: A coordinated symbolic debugger.

COMPATIBILITY: *TASM* is source code compatible with MASM and supports macros, records and structures.

Introductory Price $49
With .OBJ Capability $99

Speedware™

Include $5.00 shipping and handling. California residents add 6% Sales Tax.

Dealer inquires welcome

916-988-7426
118 Buck Circle, Box C
Sacramento, CA 95630

IBM, Microsoft trademarks of IBM Corp., Microsoft Corp.

CIRCLE 41 ON READER SERVICE CARD

ARGON UNIVERSAL SOFTWARE

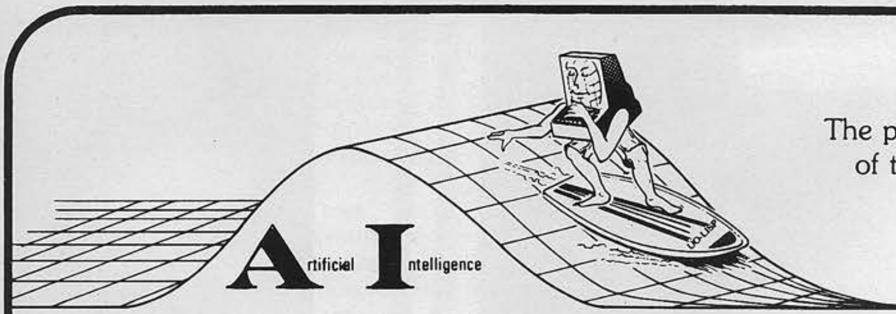
We Beat ALL Prices!

	LIST	OURS
Ascom	195	120
Crosstalk	195	120
Greenleaf Communications	185	150
Greenleaf Functions	185	150
Instant C	495	400
Lotus	495	320
Microsoft C	395	270
Pfinish	395	260
Pfix 86+	395	260
Plink 86	396	260
Pmaker	195	140
Pmate	225	145
Pre C	395	260
Run C	150	99
Wizard C	450	350

253 10th Ave.
New York, N.Y. 10001

212-989-7167

CIRCLE 5 ON READER SERVICE CARD



LISP

The preferred symbolic processing language of the Artificial Intelligence Community

catch the next micro-wave with

UO-LISP

Not "just another pretty dialect" but the most powerful implementation of LISP available in the micro market place. For the professional engineers, researchers, and educators, UO-LISP maintains the power and flexibility inherent in LISP while providing the expected functionality of mainframe LISP systems. (+) **UO-LISP steps beyond the competition and provides a real source to native code compiler.**

CPU Family	Operating System	Production System	Learn System
8086	MS-DOS	150^{00}	85^{00}
	PC-DOS	150^{00}	85^{00}
	CPM/86	*available soon*	
Z80	CPM	125^{00}	85^{00}
	TRS DOS	80^{00}	N/A

For MORE DETAIL AND TO ORDER:
Send for *FREE* brochures and order forms.

NORTHWEST COMPUTER ALGORITHMS
P.O. Box 90995, Long Beach, California 90809
(213) 426-1893

CIRCLE 39 ON READER SERVICE CARD

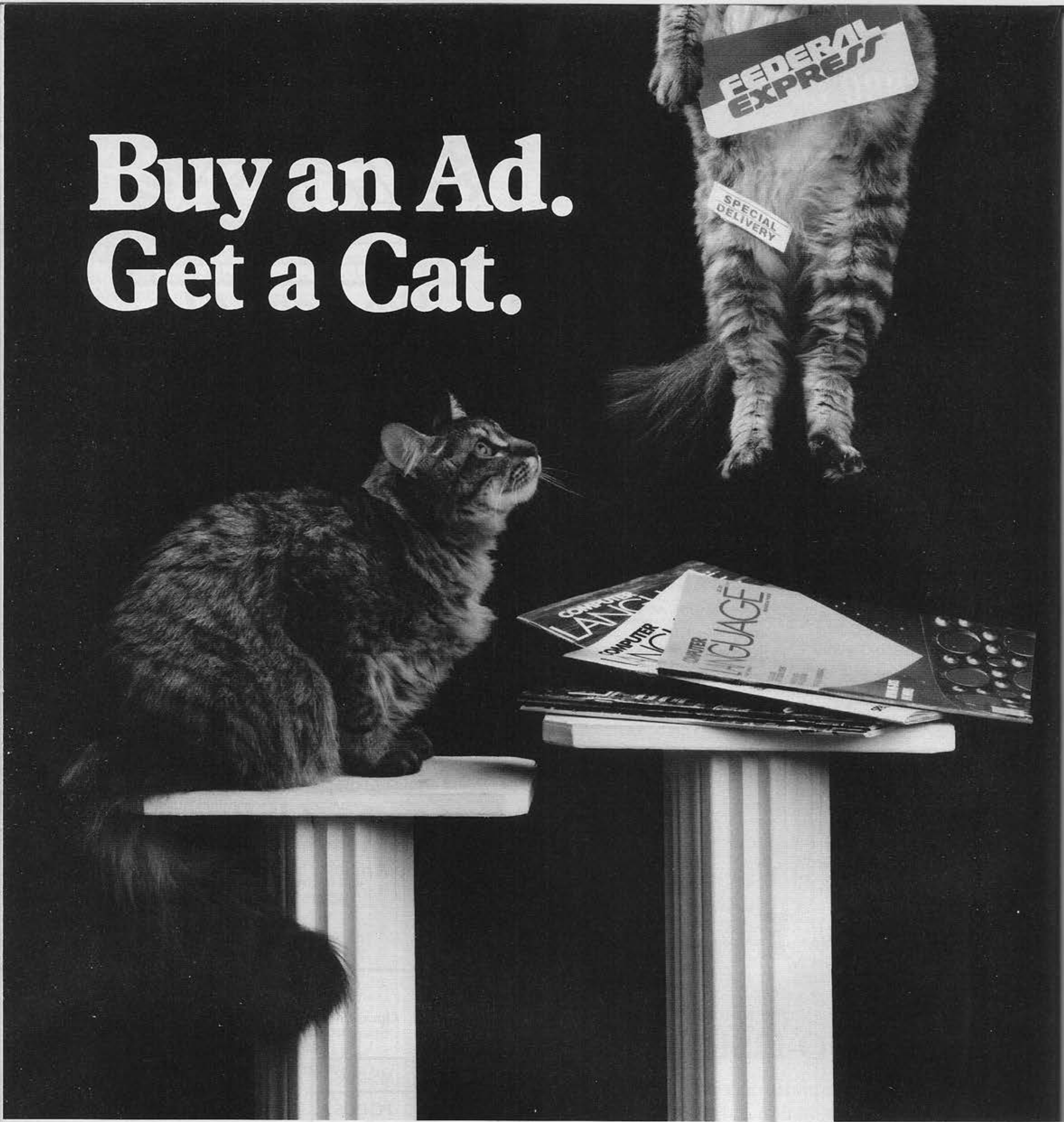

Buy an Ad.
Get a Cat.
FEDERAL EXPRESS
SPECIAL DELIVERY
COMPUTER LANGUAGE

Photo by Dow/Clement Photography

That's right, you can reach 34,000 software developers with your advertising message and get a free cat to boot. We're talking genuine felines with shots, spayed, declawed—the whole nine yards.

Just call **800-GET-A-CAT** and we'll ship immediately. Just let us know how to ship:

___UPS
___UPS Blue Label
___Overnight
___Bulk Mail

Advertise today (415) 957-9353. We're standing by ready to punch holes in boxes.

SOFTWARE REVIEWS

Knowledge systems for the IBM PC, Part II

By Ernie Tello

In this second and final installment of *COMPUTER LANGUAGE*'s coverage of current PC expert systems, we'll look at the packages falling into the following categories: intermediate-level systems, rule-oriented mathematical modeling systems, and advanced systems.

Table 1 contains general information on all products listed in Part I and Part II of this article.

INTERMEDIATE LEVEL SYSTEMS

Expert Systems ES/P Advisor

Although Expert Systems International's ES/P Advisor is a PROLOG-based tool, as opposed to an extension package like APES (reviewed in Part I), it is a fully configured consultation and development environment that does not require the user to know PROLOG. But for those who already do or are willing to learn, there are some substantial rewards waiting. Although ES/P Advisor is not merely a high-level programming language or extension to PROLOG, it allows any legal programs or functions in the PROLOG-1 dialect to be added freely to its knowledge bases.

This system seems to have been designed specifically for the creation of on-line manuals and how-to applications that take a user step by step through a process covering many different situations. It consists of two separate programs: the consultation shell, or end-user environment, and the knowledge base compiler. The run environment stands out for its well-designed use of windows and color displays and a large number of commands that allow the user to take the initiative in a consultation.

ES/P Advisor's knowledge representation language is powerful and is able to have variables assigned a value through input from a user and then be quoted in subsequent text displays. The inference engine is a backward chaining system that allows the knowledge system developer to assign goals.

Though this system is powerful, the knowledge base files are not particularly easy to read because they are structured according to the order that things will appear in rather than a logical order. Rules are not all in one section but can appear throughout the source file. I have not found this a particularly easy development environment. In many ways the KRL compiler is very strict about syntax idiosyncracies, and yet in others it allows files to compile that subsequently will not run properly. This puts the developer back in the situation of conventional programming languages, where consideration of the tools themselves diverts attention from the problem to be solved.

One of the best features of this system is the interface to the PROLOG language itself. An interface to PROLOG-1 or PROLOG-2 is also available from Expert Systems International as a separately purchased option. This feature gives ES/P Advisor an open-ended architecture that allows a skillful PROLOG programmer to embed programs into the system to augment the inference engine or interface it with external hardware.

PROLOG can be incorporated into ES/P Advisor in two ways:

- As a rule that is used to deduce a value of a knowledge base parameter
- As a pseudo-paragraph, a procedure that is invoked after a qualifying condition is satisfied.

In either case, ES/P Advisor itself does not provide any error trapping for PROLOG code. The PROLOG programs that are entered must be fully debugged beforehand or errors will crash the system. So although in principle it is not necessary to purchase the ESI PROLOG interpreter to add code to ES/P, in practice it seems essential for debugging any complex PROLOG routines.

This system is appropriate where there is adequate programming expertise available. While it is well suited to developing generic applications for resale, it is not easy enough to use to be appropriate for rapid prototyping. For programmers with expertise in PROLOG, this could be a relatively inexpensive environment in which powerful expert systems, within certain limits, could be built.

General Research TIMM-PC

The Intelligent Machine Model (TIMM) from General Research Corp. is the first knowledge engineering system available for microcomputers that handles problems where the user may not know all the necessary and vital information for a problem.

The main features of this system are a preprocessor for knowledge bases that describes the way key attribute variables behave and a partial match search routine based on the nearest neighbor algorithm. As we will see later in the discussion of the fuzzy logic of REVEAL, few real-world problems have the either/or, on/off digital logic on which computers are based. Systems that can be made to operate on the basis of a quantified similarity rather than exact congruent matching are more flexible and potentially more useful in a world where things usually change subtly and by degrees.

By incorporating a preprocessing module, TIMM allows the knowledge engineer to tell the system ahead of time how a particular variable is supposed to behave. Three types of variables are allowed: unordered, linearly ordered, or circularly ordered.

With unordered attributes, TIMM makes no assumptions about how the values can vary. In contrast, linearly ordered values assume that the permissible values will fall into some linear sequence. For example, with an attribute such as skill level, we could define the permissible values as novice, junior, senior, and expert, with each having its place in an ordered sequence. On the other hand, a variable such as month-of-delivery is considered circularly ordered because the values go through a 12-month sequence and then start over again.

Knowing in advance how an attribute can vary is one thing that gives TIMM the ability to fill in missing knowledge. For example, if it has already been given the information that junior-level and expert-level people have a certain skill, then TIMM would consider it a good assumption that senior-level people have that skill also, because senior-level falls between junior-level and expert-level in the linear sequence. By adopting this strategy on a mass scale, it is not hard to see how TIMM could frequently fill in missing information when making inferences. The other facility TIMM uses to operate when all the facts aren't available is based on the partial match algorithm mentioned earlier.

TIMM is made up of 10 separate programs that are available from a common menu. Here is the main menu as it appears on the screen:

1. Build
2. Train
3. Exercise
4. Modify
5. Inquire
6. Generalize
7. Check Consistency
8. Check Completeness
9. Compress
10. Convert Knowledge Base

The Build program allows you to enter all the choices and factors for your problem and to specify how the factors will behave. With Train you can develop the rules that the knowledge system will contain.

The example conditions are presented much as they would appear in rules. Each part of the example becomes visible as one of a cumulative list of conditions, with the choice entered last in the list. A prompt asks whether you wish to enter the examples or have the system do it for you. If you know many of the rules you want to incorporate in the system then you can enter them as examples. If you let the system choose the values for you, then it randomly picks values for each of the factors based on the possible values you have assigned them.

The Exercise program is where you actually run a knowledge system. It is here that TIMM can try to draw conclusions without having all of the values it asks for. The Modify program is the editing program that allows you to make changes in the actual constraints of the application. The Inquire program allows you to inspect the system to see what it contains. One option gives you the decision structure, meaning the entries you made in the Build program. Another allows you to see a listing of all the rules.

The Generalize program is one of the more interesting ones in the system. It uses the information so far on the system to make hypotheses to the user, who responds favorably or unfavorably and determines whether or not the idea becomes a rule. The rest of the programs are straightforward utilities. The presence of inconsistencies here does not mean logical contradictions. Perfectly viable knowledge bases can have several inconsistencies.

The main limitations of TIMM are that it only allows up to 25 end choices in a knowledge system, it has no provisions for variables or mathematical calculations, and it has no external interface. There are plans, however, to continue to extend this program. The manufacturer should be contacted for information on the latest release.

There are a number of very interesting ideas implemented in TIMM, and I for one hope the authors continue to enhance it and add all the features needed to make this a complete system with all the capabilities of the most advanced systems.

Level Five Research Insight Knowledge System 2

Level Five Research Inc.'s Insight 2 looks and feels like Insight, the $95 product by the same company that was described in Part I of this series, but it is a far more powerful system worth considering as a professional development tool.

The two products share identical end-user interfaces, but Insight 2's is not a separate, stand-alone program. Everything is integrated into a single program available from a menu. Included are a WordStar-like text editor, a data base editor, a compiler, and a run-time environment. Unfortunately, I can't say a great deal about the editors because they were not finished in the prerelease version tested.

The Ultimate Programmer's Editor
WENDIN'S XTC™

SUPER PROGRAMMERS edit in XTC to make software development a snap! Just look at these powerful features:

MULTITASKING XTC's built-in multitasking lets you run your macros in the foreground or independently in the background while you continue editing. A background process has full access to editor resources, and can be used to translate code from one language to another in REAL TIME, print files in the background, or even scan syntax while you type in code. Best of all, you can use XTC to edit source and documentation in any programming language!

COMPILE IN WINDOWS All DOS compilers and utilities can be executed from within XTC using a single keystroke. While it runs, XTC captures your compiler's output and redirects it into your text, so you can compare compiler messages with your source code ON THE SAME SCREEN. And using XTC's macro language, Turbo Pascal is literally only a keystroke away. You can use other compilers and utilities inside XTC too — like Lattice "C," Microsoft Pascal, and IBM's Basic, to name a few.

MACRO LANGUAGE XTC has the most powerful macro langague in the editing world. XTC's macros aren't just keystrokes assigned to keys; they're real programs that can be used to automatically edit source code and data files. Like any real programming language, XTC has control structures like IF THEN ELSE, WHILE DO, REPEAT UNTIL, FOR NEXT, DUPLICATE N TIMES, INDEFINITE LOOP, EXIT, and BREAK LOOP. XTC also has INTEGER, BOOLEAN, and STRING variables to hold numbers, conditions, and pieces of text.

WINDOWS & BUFFERS With XTC you can display up to 8 different files or parts of the same file on the screen at once. XTC's windows are programmable and can even be linked together to share files. XTC also has 20 other buffers that you can use to hold files and blocks of text.

INCREDIBLE EXTRAS! XTC comes with 7,000 lines of source code jam-packed onto two DSDD disks. Includes 13 modules written in Pascal, and 2 assembly libraries you can use to access the PC's screen, intercept software interrupts (like INT 21H functions), allocate and deallocate memory, and load and execute programs. It's all included FREE for your recreation and enjoyment!

- UNDO N TIMES
- REMOVE WORDSTAR HIGH BITS
- EDIT GRAPHICS DISPLAYS
- AUTOINDENTING MODE
- TAB EXPANSION/ COMPRESSION MODE
- EXTRA LONG LINES
- MACRO COMPILER
- TELEPHONE SUPPORT
- 150 PAGES OF DOCUMENTATION
- RUNS ON IBM / PC, XT, AND / AT COMPUTERS (AND TRUE COMPATIBLES)

INCLUDED FREE!
- Macro Compiler with full source code in "C"!

To get your copy of XTC now, order it over the phone — we can ship it the same day you call! Or, send in an order, just like this one:

XTC **$99.00**
Macro Compiler ...... **FREE**
Shipping, Handling, Insurance **3.50**
Want it COD? Add this **1.90**

WENDIN
BOX 266
CHENEY, WA 99004
The people who make quality software tools affordable.

ORDER HOTLINE
509/235-8088
CREDIT CARDS WELCOME!

Ada* is a registered trademark of the U.S. Department of Defense. Turbo Pascal is a trademark of Borland, Inc. XTC is a trademark of Wendin, Inc.

CIRCLE 43 ON READER SERVICE CARD

Tools That Make Your Job Easier

For PCDOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others
For CPM80 2.2/3.0 (Z80 required/64K) • 8″ SSSD, Kaypro 2/4, Osborne I SD/DD, Apple II, & others

MIX EDITOR

Programmable, Full/Split
Screen Text Processor

Introductory Offer 29⁹⁵

Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, it's easy to change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use.

MSDOS Features

Execute any DOS command or run another program from inside the editor. You can even enter DOS and then return to the editor by typing exit.

MIX C COMPILER

Full K&R Standard C Language
Unix Compatible Function Library

Introductory Offer 39⁹⁵

Complete & Standard

MIX C is a complete and standard implementation of C as defined by Kernighan and Ritchie. Coupled with a Unix compatible function library, it greatly enhances your ability to write portable programs.

The Best C Manual

MIX C is complemented by a 400 page manual that includes a tutorial. It explains all the various features of the C language. You may find it more helpful than many of the books written about C.

Fast Development

MIX C includes a fast single pass compiler and an equally fast linker. Both are executed with a simple one line command. Together they make program development a quick and easy process.

Fast Execution

The programs developed with MIX C are fast. For example, the often quoted prime number benchmark executes in a very respectable 17 seconds on a standard IBM PC.

Standard Functions

In addition to the functions described by K&R, MIX C includes the more exotic functions like *setjmp* and *longjmp*. Source code is also included.

Special Functions

MIX C provides access to your machine's specific features through BDOS and BIOS functions. The CHAIN function lets you chain from one program to another. The MSDOS version even has one function that executes any DOS command string while another executes programs and returns.

Language Features

- Data Types: char, short, int, unsigned, long, float, double (MSDOS version performs BCD arithmetic on float and double-no roundoff errors)
- Data Classes: auto, static, extern, register
- Struct, Union, Bit Fields (struct assignment supported)
- Typedef, Initialization
- All operators and macro commands are supported

30 DAY MONEY BACK GUARANTEE

Orders Only: **Call Toll Free 1-800-523-9520,** (Texas only **1-800-622-4070**)

MIX Editor ____ $29.95 + shipping ($5 USA/$10 Foreign) Texas residents add 6% sales tax
MIX C ____ $39.95 + shipping ($5 USA/$25 Foreign) Texas residents add 6% sales tax
Visa ____ MasterCard ____ Card # ________________ Exp. Date ____
COD ____ Check ____ Money Order ____ Disk Format ________
Computer ________ Operating System: MSDOS ____ PCDOS ____ CPM80 ____
Name ________________
Street ________________
City/State/Zip ________________
Country ________________
Phone ________________

MIX software
2116 E. Arapaho
Suite 363
Richardson, Tx 75081

Dealer Inquiries Welcome
Call (214) 783-6001

MSDOS is a trademark of Microsoft PCDOS is a trademark of IBM CPM80 is a trademark of Digital Research WordStar is a trademark of MicroPro L

CIRCLE 24 ON READER SERVICE CARD

As I write this, Insight 2 still has a ways to go until it is finished, but it is clear what this system will become, and it is impressive. Once again, Level Five Research is offering a lot of value for the money. So what does this higher-end package offer over the entry-level version? In addition to the production rule language, DBPAS, a built-in mini-Pascal that allows you to write programs, can be called directly from a PRL knowledge base. You can also call external programs written in other languages, but if you're willing to bite the bullet and use Pascal you have the built-in ability to pass parameters from the knowledge system to the program and back again.

With Insight 2 it is now possible to compose the actual text of questions rather than rely only on the automatic question-generation facility alone. Multi-paged text displays are provided which are activated when certain rules fire. Displays can also use bar graphs to convey their information. A goal selection facility allows more knowledgeable users to skip the preliminaries and jump directly to the subdomain of a particular problem. A *pursue* facility also is included for exploring "what if" possibilities and determining the likelihood of certain situations occurring.

The PRL in Insight 2 has been expanded to include arithmetic operations within rules. The system can use as much memory as you have in your machine. Other additions include full support of object-attribute-value triples; external programs, such as data base programs and custom programs that can be executed with the *ACTIVATE* command; knowledge bases that can be chained together; and techniques for implementing inference procedures that essentially operate like a forward chaining inference engine.

A sample dBASE II data base interface is included in the appendix of the documentation. The demo DBPAS program makes all the variable and data base record declarations. It also implements a procedure that opens a data base file, indexes through it, and lists the record information in various ways.

An *APPEND* procedure makes it possible to update the data files by adding new records. Since this is done in the DPAS language, a subset of Pascal that interfaces directly with Insight 2 knowledge bases, an entirely open-ended situation is present. Those adept at the language can do many of the things that would ordinarily be done with Pascal. But in this case, these functions may be used to integrate existing data bases with the Insight 2 knowledge engineering environment. Although this interface is clearly biased toward one programming language, much as ES/P Advisor is toward PROLOG, it offers one of the most powerful extendable environments at a reasonable price.

RULE-ORIENTED MATHEMATICAL MODELING SYSTEMS

Lotus/Software Arts TK!Solver

Most of the systems reviewed here are not overly impressive in their math capability, but TK!Solver by Lotus/Software Arts is a clear exception. This unusual package is clearly an expert arithmetic problem- and equation-solver. But what it is or can be besides is not as easy to determine.

Certainly, TK!Solver is not an expert system tool of the usual production system type, being somewhere between a spreadsheet and a symbolic math solver. But it is clearly a far cry from a mathematical expert system of the caliber of the MACSYMA system developed at the Massachusettes Institute of Technology, Cambridge, Mass. Yet the potential for TK!Solver applications by a sophisticated developer goes well beyond mere run-of-the-mill math and engineering problems.

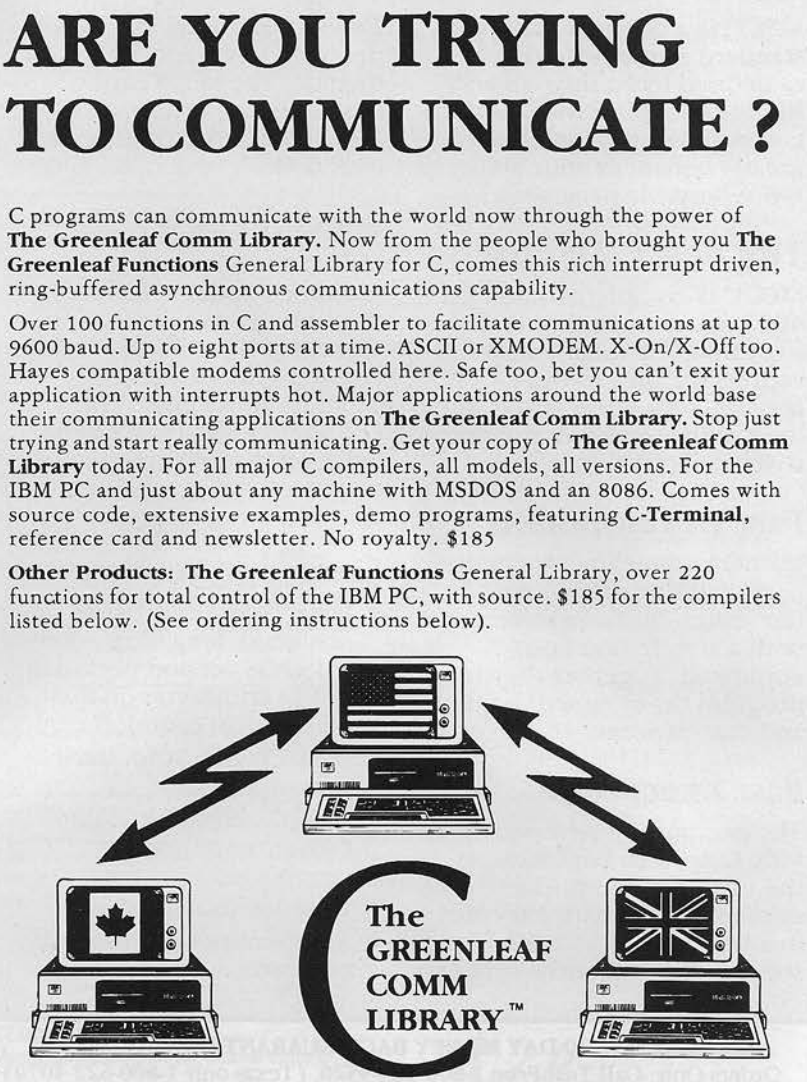
The basic environment has eight main information modules or sheets: variable, rule, list, global, unit, table, plot, and user function. Each acts as a separate window onto a data set that drives the two problem solvers: the direct solver and (in

ARE YOU TRYING TO COMMUNICATE ?

C programs can communicate with the world now through the power of **The Greenleaf Comm Library.** Now from the people who brought you **The Greenleaf Functions** General Library for C, comes this rich interrupt driven, ring-buffered asynchronous communications capability.

Over 100 functions in C and assembler to facilitate communications at up to 9600 baud. Up to eight ports at a time. ASCII or XMODEM. X-On/X-Off too. Hayes compatible modems controlled here. Safe too, bet you can't exit your application with interrupts hot. Major applications around the world base their communicating applications on **The Greenleaf Comm Library.** Stop just trying and start really communicating. Get your copy of **The Greenleaf Comm Library** today. For all major C compilers, all models, all versions. For the IBM PC and just about any machine with MSDOS and an 8086. Comes with source code, extensive examples, demo programs, featuring **C-Terminal,** reference card and newsletter. No royalty. $185

Other Products: The Greenleaf Functions General Library, over 220 functions for total control of the IBM PC, with source. $185 for the compilers listed below. (See ordering instructions below).



Specify compiler when ordering: Lattice, Microsoft, Computer Innovations, Mark Williams, or DeSmet. Add $7.00 for UPS Second Day Air (or $5.00 for ground). Texas residents add sales tax. Mastercard, VISA, check, or P.O. In stock, shipped same day.

☐General Libraries	$185
☐Comm Library	$185
☐CI186 Compiler	$349
☐Lattice C	$395
☐Mark Williams	$475

PRICES ARE SUBJECT TO
CHANGE WITHOUT NOTICE.

For Information:
214/446-8641

GREENLEAF
SOFTWARE©

2101 HICKORY DR.
CARROLLTON, TX 75006

CIRCLE 44 ON READER SERVICE CARD

tlc-lisp86

— Immediate Delivery —

Zetalisp™-like capabilities PLUS 8087 support
Smalltalk™-like class system PLUS first class instances, closures and environments
WordStar®-like display editor PLUS Lisp specific operations

ADDITIONAL FEATURES:

SOURCE FILES for Editor, Debugger and OS interface included for user customization
COMPILER for best performance of user-defined functions
CP/M, MS-DOS, PC-DOS support, Software Interrupts, full access to Memory and I/O Ports
DOCUMENTATION comprehensive Primer and Reference Manual
TLC™-Logo's MULTI-TURTLE GRAPHICS for IBM PCs

☐ **Complete System $250.**
☐ **Manual Only $20.**

THE lisp™ COMPANY (T.(L.C))

The LISP Company (T.(L.C))
430 Monterey Ave. #4
Los Gatos, California 95030
408/354-3668

Also Available: TLC™-Logo $100., and our book Thinking About TLC™ Logo $18.

Zetalisp is a trademark of Symbolics, Inc., WordStar is a registered trademark of MicroPro International Corp., Smalltalk is a trademark of Xerox Corp.

CIRCLE 65 ON READER SERVICE CARD

SPARRY BASIC-B COMPILER

1. Floating Point Math
2. Use all 640K of Memory
3. Multiple Data Segments
4. Multiple Code Segments
5. Internal ISAM File support
6. 4 Virtual Screens (Big Windows)
7. Easy Assembly Language Interface
8. Direct System Interrupt Calls
9. A Compatible BASIC Compiler

Req. PCDOS 2.00 + with 128K

Sparry Software Labs

P.O. BOX 632
MILFORD, MA 01757
617-473-5435

( ) Compiler $159
( ) Demo Disk $15

PCDOS is a Trademark of International Business Machine Corp.

CIRCLE 66 ON READER SERVICE CARD

ConIX™

NOW ONLY $79.95!

If you think you're missing out on innovative software developments because nobody is writing for CP/M™-80, take a look at us. We've adapted UNIX™ features to CP/M like never before, and with the kind of professional, quality-controlled product that you deserve. That product is none other than the critically acclaimed ConIX Operating System.

ConIX can provide any 48K+ CP/M-80 or compatible system with I/O Redirection and Pipes (uses memory or disk), perfected User Areas, Command and Overlay Path Searching, Auto Screen Paging, 8Mb Print Buffering, 22 new SysCalls, Function Keys, "Virtual" disk system, Archiver (saves over 50% disk), extensive command language, 300+ variables, 100+ commands, pull-down menu, and much more! Uses as little as 1/2K RAM! Runs with CP/M for true data and software compatibility. Installs easily without any system mods!

The ConIX package lists at $165 and has been advertised and sold internationally to many enthusiastic customers since October 1983. As a special limited offer, we've lowered the price of the complete ConIX system by 50% to only $79.95! Don't miss this opportunity to bring your 8-bit micro back into the software revolution. Order your copy of ConIX today!

Price includes manual, 8" disk, and user support. 5¼" conversions available. Contact your local dealer, or buy direct and add shipping: $4.50 UPS, $10 Canada, $25 overseas. NY residents add sales tax.

Computer Helper Industries Inc.
P.O. Box 680 Parkchester Station, NY 10462
Tel. (212) 652-1786 (for information/orders)

"We're helping your computer work better for you!"

UNIX: AT&T Bell Labs, CP/M: Digital Research, ConIX: Computer Helper Ind.

CIRCLE 10 ON READER SERVICE CARD

edward k. ream

Transform Your Programs with CPP—C Preprocessor Plus

Includes ALL features of the standard C preprocessor.
- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Conditionally include or exclude lines with #if, #ifdef and #ifndef commands.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or * are ignored.)

Fast and flexible
- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

Complete
- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

Price: $50.

Call or write today:
Edward K. Ream
1850 Summit Ave., Madison, WI 53705
(608) 231-2952

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5¼ inch disk). Send a check or money order for $50 ($60 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check in included.

case the first fails) the iterative solver.

The emphasis of this design, as opposed to that of a spreadsheet, is on seeing clearly what goes in and what comes out in the calculation of answers. No attempt is made to provide a permanent storage matrix, as with a spreadsheet. The formula, or rule, sheet is kept separate from the sheet that contains the variables and their values. Rather than trying in vain to describe in general terms how this works, it will be much easier to give an example of TK!Solver at work on an interesting problem.

The example comes from *The TK!Solver Book*, by Milos Konopasek and Sundaresan Jayaraman, the developers of the system. It gives solutions to the so-called Einstein Twins Paradox. More exactly, it is a model based on the equation for time dilation as predicted by special relativity. Listing 1 shows the rule sheet and the variable sheet.

The items under the Name column are all the variables used on the rule sheet. They appear automatically here by virtue of being entered on the rule sheet. Problems are solved by entering values in the Input column for all the variables except the one being solved for. You recalculate and TK!Solver tries to find the answer with the direct solver. If it can't, you then call on the iterative solver. As you can see, with the input and output columns on either side of the variables column, the focus is on what goes in and what comes out of the model. Among other things, TK!Solver is actually a highly interactive, input/output-oriented simulation modeler.

The rules used here are not the *if . . . then* type used in production systems but are equations or formulas that follow the variables rather than applying to the cell locations. There is no way to set up consultations with TK!Solver where the user is asked questions. And there is no querying about how or why it gives the answer it does. Someone who needs to constantly get numerical answers to complicated problems may prefer this format, because it offers a hands-on interactive style with everything clearly laid out. If the ultimate point is to get the problem solved, then the consultation paradigm may not always be as satisfactory as TK!Solver's plug-in approach.

Although TK!Solver was not specifically designed to accomplish multiple condition statements and logical reasoning, Konopasek and Jayaraman have demonstrated methods for allowing this. These techniques are esoteric but fascinating, and most importantly, they work.

A good example of this, also from *The TK!Solver Book*, is on the color coding of resistors. By typing in the color names for each of the bands on a resistor in the input cells, the application can take in these names for color combinations and spit out the value of the resistance in ohms and the manufacturer's tolerance in percents. And by backsolving, the process can be reversed, returning the color code pattern when the resistance and allowable tolerance are provided.

It is quite conceivable that someone might build a small production rule knowledge system for the color coding and decoding of resistors. But why go through all the questions and reasoning when, with a system like TK!Solver, you can just plug in the pattern and immediately get your answer? It is clear that TK!Solver is quite limited in some ways and doesn't have the tools to handle powerful symbolic processing problems. But it is good to recognize that TK!Solver's simpler input/output paradigm may be the best approach to an important class of problems.

```
Rule sheet

S  Rule
-  ----

*  Ts = Te * sqrt(1 - v^2/c^2)

-------------------------------------------------------------

Variable sheet

St  Input      Name   Output  Unit  Comment
--  -----      ----   ------  ----  -------

L              Te             yr    time on earth
    10         Ts             yr    time in the spaceship
L    0         v              m/s   spaceship velocity
    299790000  c              m/s   velocity of light
```

Listing 1.

Macintosh Programmer's Library

Affordable, Reliable Tools for Programmers

From the author of Pascal/MT+ Tardis Software presents the

Mac Programmer's Library

A collection of tools for the serious Mac programmer. Tools which save you hours of work written by an experienced Macintosh programmer. Tardis Software is committed to giving you reliable tools at a good price. Order 3 and save 10%, order all 5 and save 20%!.

The Macintosh Programmer's Library current contents:

- FastFinder - command-line oriented user-interface.
- ToolPak 1 - diff, search, dump ASM xref, librarian
- ToolPak 2 - c beautifier, tool subroutines (C source) and C xref
- C-leaner - expert system program, tells you where and how to optimize existing code
- MacMake - program maint. utility, simplifies re-builds

All are $49.95! and are AVAILABLE NOW

MC/Visa. COD's and POs add $5 for handling. Price incl. UPS ground, Blue Label add $5

Tardis
Software
2817 Sloat Road
Pebble Beach, CA
(408) 372-1722

CIRCLE 82 ON READER SERVICE CARD

Program Editing with

BRIEF™

is More Productive and Less Frustrating

because it will work YOUR way, and BRIEF elegantly integrates:

- A high-level, readable Macro Programming Language - allows full parsing or syntax analysis . . . Complete, unlimited variables, etc.
- Edit **multiple** files of **unlimited size** (2 Meg is OK)
- Multiple **Windows** on screen with different or same file, fragments, etc.
- A bona-fide **UNDO** stack (up to 300) of **all operations:** deletions, reading files, search, translate, more

FREE WITH ORDER: Best of BRIEF Macros" — includes Fortran, C, Calculator. Call before 11/30/85.

Only $195

Full Refund if not satisfied in 30 days.
CALL **800-821-2492.**

- Full **"regular expression search"** - wild cards, complex patterns
- A completely **reconfigurable keyboard**
- **Keystroke macros** - for common typing sequences
- Suspend BRIEF to execute, **exit to DOS** - run another program (like a compiler, dir, XREF, DIFF, or DEBUG) then resume BRIEF session
- **Compiler-specific** support like auto indent, syntax check, compile within BRIEF

For PC, AT, and compatibles.

Solution Systems™

335-L Washington St., Norwell, MA 02061
617-659-1571

CIRCLE 91 ON READER SERVICE CARD

PROLOG-86™

Become Familiar in One Evening

Thorough tutorials are designed to help learn the PROLOG language quickly. The interactive PROLOG-86 Interpreter gives immediate feedback. In a few hours you will begin to feel comfortable with it. In a few days you are likely to know enough to modify some of the more sophisticated sample programs.

Sample Programs are Included like:

- ■ **an EXPERT SYSTEM**
- ■ **a NATURAL LANGUAGE INTERFACE** (it generates a dBASEII "DISPLAY" command)
- ■ **a GAME** (it takes less than 1 page of PROLOG-86)

PROTOTYPE Ideas and Applications QUICKLY

Serious development of experimental systems and prototypes is practical with the full syntax of PROLOG-86. 1 or 2 pages of PROLOG is often like 10 pages in "C".

Programming Experience is not required but a logical mind is. PROLOG-86 supports the de facto STANDARD.

RECENT IMPROVEMENTS: Access to MSDOS, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

FREE with order: "Best of Prolog-86 Programs"—contest entries include: a primate expert system, an automobile expert system, a blocks world natural language system, etc. Call before November 30.

Only
$125.
Full refund if not
satisfied during
first 30 days.

Solution Systems

335-L Washington St.,
Norwell, Mass. 02061
617-659-1571
800-821-2492

CIRCLE 42 ON READER SERVICE CARD

NEW FEATURES

(Free update for our early customers!)

- Edit & Load multiple memory resident files.
- Complete 8087 assembler mnemonics.
- High level 8087 support.
  Full range transcendentals
  (tan, sin, cos, arctan, logs and exponentials)
  Data type conversion and I/O formatting.
- High level interrupt support.
  Execute Forth words from within machine code primitives.
- 80186 Assembler extensions for Tandy 2000, etc.
- Video/Graphics interface for Data General Desktop Model 10

HS/FORTH

- Fully Optimized & Tested for:
  IBM-PC IBM-XT IBM-JR
  COMPAQ EAGLE-PC-2
  TANDY 2000 CORONA
  LEADING EDGE
  (Identical version runs on almost all MSDOS compatibles!)
- Graphics & Text
  (including windowed scrolling)
- Music - foreground and background
  includes multi-tasking example
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management Support
- Full megabyte - programs or data
- Complete Assembler
  (interactive, easy to use & learn)
- Compare
  BYTE Sieve Benchmark jan 83
  HS/FORTH 47 sec BASIC 2000 sec
  w/AUTO-OPT 9 sec Assembler 5 sec
  other Forths (mostly 64k) 70-140 sec

FASTEST FORTH SYSTEM
AVAILABLE.

TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!
(TEN TIMES FASTER WHEN USING AUTO-OPT!)

HS/FORTH, complete system only: $250.

VISA Visa Mastercard
Add $10. shipping and handling

HARVARD SOFTWORKS

PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE 47 ON READER SERVICE CARD

NGS FORTH

*A FAST FORTH,
OPTIMIZED FOR THE IBM
PERSONAL COMPUTER AND
MS-DOS COMPATIBLES.*

STANDARD FEATURES
INCLUDE:

- 79 STANDARD
- DIRECT I/O ACCESS
- FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ENVIRONMENT SAVE & LOAD
- MULTI-SEGMENTED FOR LARGE APPLICATIONS
- EXTENDED ADDRESSING
- MEMORY ALLOCATION CONFIGURABLE ON-LINE
- AUTO LOAD SCREEN BOOT
- LINE & SCREEN EDITORS
- DECOMPILER AND DEBUGGING AIDS
- 8088 ASSEMBLER
- GRAPHICS & SOUND
- NGS ENHANCEMENTS
- DETAILED MANUAL
- INEXPENSIVE UPGRADES
- NGS USER NEWSLETTER

*A COMPLETE FORTH
DEVELOPMENT SYSTEM.*

PRICES START AT $70

NEW◆HP-150 & HP-110
VERSIONS AVAILABLE

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

CIRCLE 61 ON READER SERVICE CARD

The last TK!Solver application example I will discuss is one that employs logical inference to diagnose automobile problems. Here new user functions are added to TK!Solver by assigning "true" to 1 and "false" to 0 and using numeric means to develop the relations for all the logical operators. In this example, input values for the variables of various conditions of parts of a car are entered and then other conditions are concluded. In one case, the car was described as fit at the same time that two tires were noted as punctured. An error message was displayed.

Though this is a toy example, it shows that, similar to Expert Ease, TK!Solver can do input/output modeling exclusively with qualitative attributes. However, since no questions are generated with this system, there is no guarantee that the user will put in just the relevant number and choice of input values to fire the responses that will help. The more complex the problem, the more the consultation paradigm is justified in providing users with guidance and explanations to stop them from getting lost.

Based on actual examples, I have seen that the following types of problems can be solved with TK!Solver: nonlinear equations for two unknowns, linear regression, curvilinear regression, random number generation, numerical differentiation and integration, and some symbolic differentiation and optimization.

The most decisive advantage TK!Solver has over spreadsheets is that, in principle, it can solve for any variable immediately. To do this with a spreadsheet, extensive editing and even an entire new spreadsheet might be required. Compared with BASIC, or nearly any other programming language, TK!Solver is far quicker and easier to use. Also, like production systems, it is very easy to modify and expand. It is completely modular in the sense that a large number of independent models can be merged into action at once.

Generally, TK!Solver is easy to use, but a user has to know what he or she wants and how to get it. It seems to be the kind of system where the same person would typically be both user and developer. Provisions for a DIF file interchange ensure compatibility with a broad range of existing software. Though not a general problem solver, TK!Solver may be the closest thing to it on microcomputers. This system is a rare combination of power and ease of use.

McDonnell-Douglas REVEAL

So far very few products really make a serious attempt to bridge the gap between expert system technology and the more familiar business information and decision support systems. But REVEAL, by McDonnell-Douglas, does this in some very interesting ways, crossing over several application boundaries at the same time. A powerful relational data base system is combined with an approximate reasoning facility, based on fuzzy sets, that incorporates aspects of natural language processing, expert systems, decision support systems, and modeling and simulation. While this may sound like quite an accomplishment, it is not too hard to see the sense in which it is true.

At first glance, it might seem that the last thing we want is fuzzy reasoning. However, digital technology frequently imposes the false precision of an all-or-nothing logic where it simply doesn't belong. The best example of this is in specifying quantitative criteria for data base queries, production rules, discrimination and transition networks, etc. There are times when something is just over the line on several counts, but the combination makes it a highly attractive option. Conventional software usually misses this entirely, whereas a competent manager or analyst seldom would. Trying to make up for this with conventional production rules can get very cumbersome, but fuzzy sets provide an easier way out.

With fuzzy sets, true or false is no longer an on-off affair but a continuous function with differing numeric values. The numbers measure the degree to which something is true—not the probability that they are true but the extent to which they are true. The idea is that some options are not 100% sound (100% true) and some not 0% sound (0% true). It is often true—to at least a certain extent—that a given option is a sound one.

With a system like REVEAL, if a rule or query states "All the options with a reasonable profit," the word "reasonable" is defined as a relatively continuous function. The records involved are assigned a value representing the degree to which they satisfy the criterion. With this approach, all the usual set relations—such as intersection, union, and so on— can be given a functional meaning. A key relation, though, is that one set logically implies another.

To provide for saying something like "If a company has very high profits, then it has very high sales," REVEAL uses the maximum-minimum rule of compositional inference to implement its fuzzy implication. This means following two provisions: the degree of truth of the consequent set (in this case high profits) cannot be of a higher value at any point than that of the antecedent set (in this case high sales); and the value selected from the consequent set to be the implied amount is the most possible or maximum value for any given antecedent value. So in the column of values of the sales corresponding to a given profit value, the one selected is the largest one. One difficulty with this approach is that since the logical relation is replaced with a mathematical curve, there is no way to portray that profits imply sales while sales do not necessarily imply profits.

REVEAL's architecture is complex but well integrated. At any given time, the full environment is referred to as the context. This context contains four things:

- Data tables that form the raw material on which the rest of the logic will operate
- The data dictionary, which consists of user-defined constants and variables
- A program in the REVEAL language which can be a normal algorithmic procedure or a policy consisting of approximate reasoning based on fuzzy set logic
- A command file with control parameters which provides a given sequence of operations that occur as a batch operation.

The REVEAL user environment operates like an interpreter with a large number of command modes. The main parts of this environment are the editor (a fairly powerful command-driven line editor), the programming language, the command file processor, the data handler, the approximate reasoning facility, and the report and graph generator.

Model programs, label sets, and command files are edited in the editing mode. Included in the editing subcommands are the usual editing functions, as well as a *PARSE* command that toggles syntax checking on and off, a *STACK* buffer for temporarily clearing the editor workspace, and *VERIFY*, which switches verification messages on or off. I did not see any character or block editing functions in the editor, however.

REVEAL's programming language is block-structured and reflects its FORTRAN heritage. However, as previously mentioned, this language is interpreted. In addition to the usual string and constant identifiers and logical, arithmetic, and inequality operators, REVEAL includes built-in functions and series identifiers for referring to the rows in data arrays. Branching to labels, repeat looping, and conditional expressions are also supported. Special functions include data handling, mathematics, finance, string variables, integer arithmetic, data moving, and external file operations. As you can see, REVEAL is a very complete language that is quite suitable for providing data manipulation, modeling, and calculation.

Besides the regular programming language, REVEAL has command file parameters and policy statements that can incorporate the fuzzy sets. Command files automate model processing by batching operations that would ordinarily be done by commands from the keyboard. Branching within a command file and nesting with other command files also are supported.

To use the policy facility, you first declare a vocabulary. Each vocabulary set is identified by its name. As soon as a set is declared it contains a default of hedges and noise words and one qualifier, named *TRUE*. The default hedge words are *ABOUT, ABOVE, BELOW, NOT, QUITE*, and *VERY*, with the synonyms *AROUND, NEAR; MORE THAN;* and *LESS THAN*.

Each of these default words represents a continuous fuzzy truth function that acts as a modifier on the truth curve of what comes after it. It is possible to nest hedge words indefinitely, as in *VERY VERY, NOT QUITE*, and even *QUITE LOW BUT NOT VERY LOW*.

To help construct statements that read like colloquial English, noise words are also defined in a vocabulary but are treated as if absent. The default noise words for each new vocabulary are *MY YOUR OUR THEIR*, *HIS HER ITS*, *A AN HE*, *THIS THAT THESE THOSE*, *SHOULD WOULD COULD*, *MIGHT MAY MUST*, and *THAN TO*.

In addition to the hedges and noise words, new fuzzy qualifiers such as *HIGH, LOW, STRONG, WEAK*, and so on can be defined with custom truth functions that allow policy statements which reflect both the way we speak and think and the contours of real situations. Resulting policy statements would include statements such as:

```
IF PROFITS*.5 IS LOW THEN COSTS
    SHOULD BE SOMEWHAT
    LOWER
IF PROFITS IS NOT LOW THEN PRICE
    SHOULD BE MUCH HIGHER
```

Once the statements that define a policy are formulated, they can be applied to a given model and can be evaluated by executing the *APPLY* command from within a given model or from the command level. In this way, sensitivity analysis, modeling, and forecasting can all be done by forming policies and command processing files that schedule processing operations.

For many problems, particularly in the business world, the REVEAL environment provides the net result of a relational data base and equation modeling with fuzzy sets and a natural language front end with far more efficiency than if these things were done literally. In a sense, it is such a direct approach to solving a certain range of problems that it might be unnecessary to use a more unwieldy environment when REVEAL can do the job. For a broad range of knowledge-based business problems requiring a combination of both numerical and symbolic processing, REVEAL is an attractive option to consider.

ADVANCED SYSTEMS

Software Architecture & Engineering KES

Knowledge Engineering System (KES) from Software Architecture &

FOR TRS-80 MODELS 1, 3, 4, 4P IBM PC/XT, AT&T 6300, ETC.

THE COMPLETE FORTH GETS A MAJOR UPDATE: MMSFORTH V2.4

- A total software environment: custom drivers for printer, video and keyboard improve speed and flexibility. (New TRS-80 M.4 version, too!)
- Common SYS format gives you a big 395K (195K single-sided) per disk, plus a boot track!
- Common wordset (79-Standard plus MMSFORTH extensions) on all supported computers.
- Common and powerful applications programs available (most with MMSFORTH source code) so you can use them compatibly (with the same data disks) across all supported computers.
- Very fast compile speeds and advanced program development environment.
- A fantastic full-screen Forth Editor: Auto-Find (or -Replace) any word (forward or back), compare or Pairs-Edit any two ranges of blocks, much more.
- Temporary dictionary areas.
- QUANs, VECTs, vectored I/O, and many more of the latest high-performance Forth constructs.
- Manual and demo programs are bigger and better than ever!
- Same thorough support: Users Newsletter, User Groups worldwide, telephone tips. Full consulting services.
- Personal Licensing (one person on one computer) is standard. Corporate Site Licensing and Bulk Distribution Licensing available to professional users.

MMSFORTH

IT'S BETTER THAN EVER.

The total software environment for IBM PC/XT, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):

MMSFORTH V2.4 System Disk	**$179.95**

(TRS-80 Model 1 requires lowercase, DDEN, 1 40-track drive.)

- Personal License (additional modules):

FORTHCOM communications module	**$ 49.95**
UTILITIES	**49.95**
GAMES	**39.95**
EXPERT-2 expert system	**69.95**
DATAHANDLER	**59.95**
DATAHANDLER-PLUS (PC only, 128K req.)	**99.95**
FORTHWRITE word processor	**99.95**

- Corporate Site License Extensions . . . . **from $1,000**
- Bulk Distribution . . . **from $500/50 units.**
- Some recommended Forth books:

STARTING FORTH (programming)	**19.95**
THINKING FORTH (technique)	**15.95**
BEGINNING FORTH (re MMSFORTH)	**16.95**

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

CIRCLE 20 ON READER SERVICE CARD

Engineering Inc. is one of the few LISP-based systems currently available for PCs. KES was originally released on the DEC VAX running under UNIX, the CDC CYBER, and the Apollo workstation. It dates back to a system developed at the University of Maryland by James Reggia and Barry Perricone. The PC version is ported to IQLISP, a subset of the INTERLISP dialect. One of the advantages of having the LISP interpreter is that experienced programmers can add to the system the features they consider important.

KES is really three separate programs: KES.PS, KES.BAYES, and KES.HT. Each has a different type of inference engine and can develop entirely different expert systems. One of the most attractive features of this package is the ability of any of the three models to call on the others as often as needed. This makes KES the first hybrid system available for microcomputers. I have not tested this feature, however, and cannot guarantee that it is usable in the PC environment.

One of the more substantial advantages of the KES environment is the powerful array of commands available to the end user. *Justify* is the main command used to ask the system to explain its results. As with many of the other commands, options are specified by what is typed immediately after issuing the command. You can ask the system to justify either the main result, other particular results, or all of the values that have an attribute of the same name. For example, you might type:

```
justify problem = dirty carburetor
```

in an auto repair application to ask KES to provide the basis for its result. In this case, KES might respond:

```
reasons for this statement:
    age of car = 7
    type of noise = backfire (by rule:
        DIRT 1.0 )
```

Further information can then be requested by the general purpose *display* command. In this case, you could make use of the *display <rule>* option by typing *display DIRT*, which would show the rule that was used to arrive at the conclusion.

Another area where KES shines is in math. In addition to standard inequality and arithmetic operations, including exponentiation, KES also has absolute value, common and natural logarithms, and sine and cosine functions that can be calculation attachments to attributes.

KES.PS is the system's rule production module. This is a fairly standard backtracking inference engine. A typical knowledge base using this module would include a section in which the attributes are declared, a section for stating the rules, and an actions section where the goals that guide rule processing are stated. Text messages can be provided at any stage of a consultation.

The KES.BAYES module incorporates an inference procedure for statistical pattern classification that is based on the well-known Bayes' theorem. It is appropriate for problems where there is a large amount of data that can be expressed in the form of probabilities, such as when available experts, based on their experience, can provide knowledge about the likelihood of various states of affairs. The inference engine first asks for all the input attribute values. Then, using the probabilities in its knowledge base, it calculates the probability of possible outcomes by plugging values into the Bayesian probability equation. The output of a typical consultation will consist of the formatted printing of probability values for various outcomes.

This type of inference module clearly is applicable to problems where such probabilistic knowledge base data is attainable and it is appropriate for the outcome to be expressed in probabilistic terms. This type of inference mechanism is often associated with the PROSPECTOR system, where a similar approach was used for estimating the probability of valuable mineral deposits.

The KES.HT module, which uses frame representation and a hypothesis and test inference mechanism, is the most

Instant-C™:
The Best Value
In C Programming Tools

The edit-compile-link-test-debug cycle that takes tens of minutes with compilers and linkers is only seconds with the *Instant-C* interpreter. Yet it runs your programs **50 to 500 times faster** than conventional C interpreters! You get the best of both compilers **and** interpreters. Only *Instant-C* is a complete, integrated environment for creating, testing, and running your programs.

Instant-C gives you **all** of these proven capabilities in one tightly integrated package:

interpreter—*Instant-C* runs your programs faster than some compilers; has direct execution; full K&R
compiler—*Instant-C* can make stand-alone programs
full-screen language editor—shows syntax errors with cursor set to trouble spot
C source debugger—single-step, breakpoints, stack trace, more
run-time checker—validates pointer refs, array bounds, more
C source formatter—save editing time, find logical flaws
standard library with source—for best portability
linker—work with multiple source modules
Lint—extensive compile-time validation

The cheapest available examples of these tools would cost $800 (and they don't even work together). You could spend close to $3000 to get the best product of each kind, but you'd have ten times the complexity, filling megabytes of disk. *Instant-C* is faster: it performs these functions automatically. *Instant-C* is far more than the sum of its parts.

Instant-C is all of these capabilities in one package, fits on a single floppy disk, is full K&R; works on IBM PC's, compatibles, and others under DOS or CP/M-86. It costs only $495.

Instant-C is the best value in C programming tools. Guaranteed, or your money back for any reason in first 31 days.

Rational
Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, MA 01760

Instant-C is a trademark of Rational Systems, Inc.

CIRCLE 72 ON READER SERVICE CARD

FREE!

For free information on products advertised in COMPUTER LANGUAGE circle the numbers corresponding to the advertisements you are interested in. Use the Advertiser Index for easy reference.

FREE!

FREE INFORMATION

Name ____________________

Company ____________________

Address ____________________

City, State, Zip ____________________

County ____________________

Telephone number ____________________

I obtained this issue through:
☐ Subscription ☐ Passed on by associate
☐ Computer Store ☐ Other __________
☐ Retail outlet

Comments ____________________

Circle numbers for which you desire information.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Attn: Reader Service Dept.

August issue. Not good if mailed after December 31, 1985.

2/8

FREE INFORMATION

Name ____________________

Company ____________________

Address ____________________

City, State, Zip ____________________

County ____________________

Telephone number ____________________

I obtained this issue through:
☐ Subscription ☐ Passed on by associated
☐ Computer Store ☐ Other __________
☐ Retail outlet

Comments ____________________

Circle numbers for which you desire information.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150

Attn: Reader Service Dept.

August issue. Not good if mailed after December 31, 1985.

2/8

ORDER COMPUTER LANGUAGE BACK ISSUES WHILE THEY LAST!

COMPUTER LANGUAGE
COMPUTER LANGUAGE
COMPUTER LANGUAGE
PORTABILITY IN C
SPECIAL C ISSUE
NIKLAUS WIRTH AND DONALD KNUTH

Only a limited quantity of magazines are available, so order today. To receive your back issues, just fill out this coupon and mail it back with a check for $5.00 per issue.

Premier ____ copies x $5.00 = $____
Oct. '84 ____ copies x $5.00 = $____
Nov. '84 ____ copies x $5.00 = $____
Dec. '84 ____ copies x $5.00 = $____
Jan. '85 ____ copies x $5.00 = $____
Feb. '85 ____ copies x $5.00 = $____
Mar. '85 ____ copies x $5.00 = $____
Apr. '85 ____ copies x $5.00 = $____
May '85 ____ copies x $5.00 = $____
Jun. '85 ____ copies x $5.00 = $____
July '85 ____ copies x $5.00 = $____
Aug. '85 ____ copies x $5.00 = $____
Foreign orders: Add $3.00 for airmail. Total = $____

LIMITED QUANTITIES AVAILABLE. ORDER TODAY!

Name ____________________

Company ____________________

Address ____________________

City, State, Zip ____________________

Send payment and coupon to:
Computer Language Back Issues, 131 Townsend St., San Francisco, CA 94107

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD
FIRST CLASS PERMIT NO. 27346 PHILADELPHIA, PA USA

POSTAGE WILL BE PAID BY

COMPUTER
LANGUAGE

P.O. BOX 11747
PHILADELPHIA, PA 19101

Return this order card with your check to:

COMPUTER LANGUAGE
Back Issues
131 Townsend Street
San Francisco, CA 94107

Back issues must be prepaid.

PROGRAMMER DEVELOPMENT TOOLS

TURBO PASCAL AND UTILITIES

	List	Ours
Turbo PASCAL *ver 3.0* by Borland Int'l . . . . *Sale*	70	**49**
Turbo PASCAL *w/8087 or BCD* . . . . *Sale*	110	**89**
Turbo PASCAL *w/8087 & BCD* . . . . *Sale*	125	**99**
Btrieve by Softcraft	250	**199**
ESP for Pascal by Bellesoft	Call	**Call**
MetaWindows *for Turbo PASCAL* by MetaGraphics	55	**49**
Multi-Halo Graphics by Media Cybernetics	250	**199**
Screen Sculptor by Software Bottling	125	**109**
Turbo ASYNCH by Blaise Computing	100	**89**
Turbo GRAPHICS TOOLBOX by Borland Int'l.	55	**49**
Turbo POWER TOOLS by Blaise Computing . *New*	100	**89**
Turbo TOOLBOX by Borland Int'l	55	**49**
Turbo TUTOR by Borland Int'l	35	**29**
TurboPower Util w/source by TurboPower Sftwr	95	**89**
XTC Text Editor by Wendin	99	**89**

BASIC LANGUAGE

BetterBASIC by Summit Software	200	**169**
8087 Math Support	99	**89**
Run-time Module	250	**239**
Professional BASIC by Morgan Computing	99	**89**
8087 Math Support	50	**47**
True Basic from Addison-Wesley . . . . *New*	150	**129**
Run-time Module	500	**459**

OTHER LANGUAGES

8088 Assembler w/Z-80 Translator by 2500 AD	100	**89**
APL*PLUS/PC by STSC	595	**449**
Golden Common LISP by Gold Hill	495	**Call**
Janus/ADA by R&R Software	900	**699**
MASM-86 ver 3.0 *w/utilities* by Microsoft	150	**109**
Modula-2/86 by Logitech	495	**439**
MS Fortran 3.3 . . . . *Links with MS C 3.0*	350	**239**
MS Pascal 3.3 . . . . *Links with MS C 3.0*	300	**219**
Prolog VMS by Automata Design Associates *New*	100	**Call**
Prolog VML by Automata Design Associates *New*	300	**Call**
Prolog VMA by Automata Design Associates *New*	500	**Call**
Prolog-86 by Solution Systems	125	**Call**
RM/Fortran by Ryan-McFarland	595	**439**

OTHER LANGUAGE SUPPORT PRODUCTS

APL2C by Lauer Software . . . . . *Interfaces APL to C*	150	**139**
Btrieve By SoftCraft	250	**199**
Blaise Tools *for Pascal*	Call	**Call**
FORTRAN Libraries by Alpha Computer Service	Call	**Call**
Sci Subroutine Lib *for Fortran or Basic*	175	**139**

OTHER PRODUCTS

Advanced Trace-86 by Morgan Computing	175	**149**
Codesmith-86 Debugger by Visual Age	145	**129**
Polytron Products . . . . *We Carry a Full Line*	Call	**Call**
Profiler by DWB Associates	125	**89**
Rtrieve by Softcraft	85	**79**
Xtrieve by Softcraft	195	**169**

Periscope Symbolic Debugger
by Data Base Decisions

Write-protect memory board and breakout switch allows instant recovery from runaway code. Provides on-line help, windowing, extensive breakpoints, dual monitor support and more.

List Price $295 Our Price **$269**

PHOENIX PRODUCTS

SUMMERTIME SALE!

In stock and ready for immediate shipping.

Pasm86 *Macro Assembler*	295	**195**
Pfinish *Performance Analyzer*	395	**269**
Pfix-86 Plus *Symbolic Debugger for Plink-86*	395	**269**
Plink-86 *Overlay Linker*	395	**269**
Pmate *Macro Text Editor*	225	**145**
Pmaker *Program Development Manager*	195	**119**
Pre-C *Lint Utility for C*	395	**269**

Dealer Inquiries Invited

C LANGUAGE

C-terp *C Interpreter* by Gimpel Software	300	**269**
C-terp *demo system* . . . . *Applied to purchase*	45	**45**
Computer Innovations C-86 Compiler	395	**299**
DeSmet C Compiler with Debugger	159	**145**
Instant C by Rational Systems . . . . *Sale*	500	**399**
Lattice C Compiler from Lattice	500	**349**
Lattice C from Lifeboat . . . . *Ltd Qty Special*	500	**275**
Mark Williams MWC-86 w/Source Debugger . *Sale*	495	**379**
Safe C Standalone Interpreter by Catalytix	400	**Call**
Wizard C Compiler by Wizard Systems . . . . . . *Sale*	450	**359**
Xenix Development System by SCO	1350	**1099**

MACINTOSH C COMPILERS

DeSmet/Ouye C Compiler . . . . *New*	150	**139**
Mac C by Consulair . . . . *Call for Details*	Call	**Call**
Megamax C *compiler for Macintosh*	295	**239**

MICROSOFT C 3.0 AND UTILITIES

Microsoft C Compiler version 3.0 . . . . *Sale*	395	**259**
Blaise C Tools	125	**109**
Blaise C Tools 2	100	**89**
C-terp by Gimpel Software	300	**269**
C Utility Library by Essential Software	185	**139**
Greenleaf C Functions Library ver 3.0	185	**139**
Greenleaf Comm Library	185	**139**
The HAMMER by OES . . . . *New*	195	**179**
Multi-Halo Graphics by Media Cybernetics	250	**199**
PANEL *Screen Designer ver. 6.0* by Roundhill	295	**234**
C Power Paks from Software Horizons	Call	**Call**
Windows for C by Creative Solutions	195	**139**

C UTILITIES

Basic_C Library by C Source	175	**139**
Blaise C Tools	125	**109**
Blaise C Tools 2	100	**89**
Btrieve by SoftCraft	250	**199**
C Power Paks From Software Horizons	Call	**Call**
c-tree by FairCom . . . . *New version*	395	**359**
C Utility Library by Essential Software	185	**139**
ESP for C by Bellesoft	Call	**Call**
GraphiC by Scientific Endeavors . . . . . *New version*	250	**209**
Greenleaf C Functions Library ver 3.0	185	**139**
Greenleaf Comm Library	185	**139**
Multi-Halo Graphics by Media Cybernetics	250	**199**
PANEL *Screen Designer ver. 6.0* by Roundhill	295	**234**
PC Lint by Gimpel Software	100	**89**
Safe C Dynamic Profiler by Catalytix	150	**Call**
Safe C Runtime Analyzer by Catalytix	400	**Call**
Scientific Subroutine Lib *for C* by Peerless	175	**139**
Windows For C by Creative Solutions	195	**139**

The HAMMER C Library by OES

This excellent new C library is designed for creating end-user interfaces. There are functions for creating 123-like menus, managing the screen, creating input forms, prompting for inputs and more. No royalties and includes source code.

List Price $195 Our Price **$179**

LATTICE PRODUCTS

All products receive support and updates directly from Lattice. Lattice C is in stock and ready for shipment.

Lattice C Compiler from Lattice	500	**349**
C-Food Smorgasbord	150	**119**
C-Sprite *Program Debugger*	175	**149**
Curses *Screen Manager*	125	**109**
dBC *C Interface for dBase II or III Files*	250	**209**
dBC *with source code*	500	**439**
LMK *Make Facility*	195	**159**
Text Mgmt Utils *(GREP/DIFF/ED/WC/Extract/Build)*	120	**105**

TEXT EDITORS

Brief By Solution Systems	195	**Call**
Epsilon *Emacs-like Text Editor* by Lugaru	195	**179**
ESP for C *or* **Pascal** by Bellesoft	Call	**Call**
ESP for C *and* **Pascal** by Bellesoft	Call	**Call**
Vedit by Compuview	Call	**Call**
XTC Text Editor By Wendin	99	**89**

Programmer's Connection

136 Sunnyside Street Hartville, Ohio 44632 (216) 877-3781 (In Ohio)

U.S.; **1-800-336-1166** Canada ; **1-800-225-1166**

Call For Our Catalog

Account is charged when order is shipped
Prices are subject to change without notice

CIRCLE 17 ON READER SERVICE CARD

interesting and unique of the three KES modules. Since it is the only one of its kind I have seen so far on a microcomputer, it is worth elaborating in some detail. This knowledge system module is devoted exclusively to knowledge bases that diagnose such problems as equipment malfunctions and medical disorders.

The main ingredients of the KES.HT knowledge bases are: outcomes, called disorders; attributes, which are divided into manifestations (the undesirable symptoms which should not be occurring); and setting conditions, which also fix the situation but are not necessarily an indication that something is wrong. These attributes are represented in the knowledge bases by a frame template, as in this spill detection example from a sample system:

```
Name: type of spill
Is : attribute
Type: mlt
Marked: evoking
Synonyms:
    contaminants
inferred: yes
possible values:
    sulfuric acid
    carbonic acid
    chromogen R23
    hydroxyluminum
current value:
    sulfuric acid <m>
    carbonic acid <m>
    chromogen R23 <m>
    hydroxyluminum <m>
External: *none*
```

The hypothesis and test module perform their inferencing by following this procedure:

1. Get a symptom attribute from the user as suggested by the current FOCUS.
2. Retrieve the relevant causes from the knowledge base.
3. Reset the SCOPE to reflect the new input.
4. Reset the FOCUS to cover the new symptoms.
5. Repeat until no further symptom attributes are needed.
6. Assign a rating to each possible explanation.
7. A final score for each explanatory attribute is calculated on the basis of its setting score and match score. This is converted to a verbal statement of its probability of being the cause.

Because of the possibility of developing complex architectures that incorporate all three of the inference mechanisms in a single expert system, KES offers some powerful advantages over other currently available tools for PCs. The main drawback of this system is its huge size, stemming from the fact that it was ported to the PC from a minicomputer implementation. I would not suggest running it on anything less than an IBM PC or AT with at least

OPS5+—An AI development tool

Dallas, Texas-based Artelligence Inc.'s OPS5+ is a production quality implementation of OPS5, the rule-based language used to write R1 (XCON), the first commercially successful artificial intelligence program. Written in C, OPS5+ is a high-level language designed to implement forward-chaining production systems.

This enhanced version of OPS5 runs on the IBM PC or XT and compatibles such as COMPAQ. It supports up to 1,500 production rules on a 640K IBM PC and requires 256K of memory. Window-based application development tools are utilized, and the multiple windows allow separate types of information to be kept in separate areas of the screen. The source code and interactions (program I/O) are handled in the large window, while informative messages, the system state, and the command menu appear in surrounding smaller windows.

OPS5+ utilizes mouse/menu interaction for greater development speed and ease of use. In order to use the full development environment, IBM color-card or Hercules graphics and a Mouse Systems or VisiCorp mouse are needed. All user routines written in C are supported, as is full IEEE 32-bit floating-point math, with 64-bit calculation accuracy. *BUILD* action and a *BACK* command (which allows the user to back up for up to 32 steps) also are supported.

OPS5+ combines the flexibility and features of interpreted versions of OPS5 with the speed of a compiler. An example of this system's extended capabilities can be seen in the *PAUSE* command. Invoking *PAUSE* allows the user to temporarily leave OPS5+ and edit another program or use a feature of the operating system and then return to OPS5+ with no loss of state.

The documentation for this product includes descriptions of OPS5 and the OPS5+ system and several case studies. In addition to consulting and maintenance services, Artelligence also offers user seminars and has the hardware to support graphics-based development systems. OPS5+ is available to the public for $3,000 and to educational institutions for $960.

OPS5+ was not yet available for review at press time; a midsummer release was planned. The product information provided in this sidebar was supplied by the manufacturer.

By Kathy Kincade

DeSmet C

8086/8088 Development Package $109

FULL DEVELOPMENT PACKAGE
- Full K&R C Compiler
- Assembler, Linker & Librarian
- Full Screen Editor
- Execution Profiler
- Complete STDIO Library (>120 Func)

Automatic DOS 1.X/2.X SUPPORT

BOTH 8087 AND S/W FLOATING POINT

OVERLAYS

OUTSTANDING PERFORMANCE
- First and Second in AUG '83 BYTE benchmarks

SYMBOLIC DEBUGGER $50
- Examine & change variables by name using C expressions
- Flip between debug and display screen
- Display C source during execution
- Set multiple breakpoints by function or line number

DOS LINK SUPPORT $35
- Uses DOS .OBJ Format
- LINKs with DOS ASM
- Uses Lattice® naming conventions

Check: ☐ Dev. Pkg (109)
☐ Debugger (50)
☐ DOS Link Supt. (35)

SHIP TO: ____________

____________ ZIP ____

CWARE CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface on IBM format disks. Shipping included in price. California residents add sales tax. Canada shipping add $5, elsewhere add $15. Checks must be on US Bank and in US Dollars. Call 9 a.m. – 1 p.m. to CHARGE by VISA/MC/AMEX.

Street Address: 505 W. Olive, #767, (94086)

CIRCLE 11 ON READER SERVICE CARD

General information

	Statistics			User interface				Knowledge structure								Development aids			
Manufacturer and product	**Memory required**	**Memory used**	**Number of rules**	**? generation**	**Explanation**	**Windows**	**Facts[4]**	**Variables**	**Multivariables**	**Math functions**	**Certainty factors**	**Text animation**	**Example driven**	**Extendable**	**Chaining**	**Rule editor**	**Compiler**	**Trace**	**External interface**
Decision modeling software																			
Decision Support Software Inc. Expert Choice	256K	all	N/A	no	no	no	N/A	yes	yes	no	yes	yes	yes	yes	N/A	N/A	N/A	N/A	N/A
Lightyear Inc. Lightyear	192K	most	500	no	yes	yes	N/A	yes	yes	via Lotus	no	no	no	yes	N/A	yes	yes	no	no
Extension packages																			
Programming Logic Systems APES	128K	40K	1,200	yes	yes	yes	yes	yes	yes	yes	no	yes	no	yes	yes	yes	no	yes	yes
Mountain View Press Inc. Expert-2	64K	24K	120	yes	yes	no	no	yes	yes	yes	no	no	yes	yes	yes	yes	no	no	yes
Rule induction systems																			
Expert Systems Inc. Expert Ease	128K	all	255[1]	yes	no	no	yes	yes	no	no	no	no	yes	no	yes	no	no	no	no
KDS Corp. KDS	512K	some	16,000	yes	yes	yes	yes	no	no	N/A	yes	yes	yes	yes	yes	yes	yes	no	yes
Small production system tools																			
Level Five Research Inc. Insight 1	128K	all	2,000	yes	yes	no	no	yes	no	no	yes	yes	no	no	no	no	yes	yes	no
EXSYS Inc. EXSYS	256K	all	unlimited[2]	yes	yes	yes	no	yes	yes	yes	yes	no	no	no	yes	yes	no	yes	yes
Intermediate level systems																			
Expert Systems International ES/P Advisor	256K	some	unlimited[3]	yes	yes	yes	yes	yes	yes	yes	no	yes	no	yes	yes	no	yes	yes	yes
General Research Corp. TIMM-PC	640K	all	500	yes	yes	no	yes	yes	yes	no	yes	no	yes	yes	yes	yes	yes	yes	yes
Level Five Research Inc. Insight 2	448K	all	2,047	yes	yes	no	yes	yes	yes	yes	yes	yes	no	yes	yes	no	yes	yes	yes
Rule oriented mathematical modeling systems																			
Lotus/Software Arts TK!Solver	128K	66K	32,000	N/A	yes	yes	yes	yes	yes	yes	no	N/A	yes	yes	yes	yes	yes	yes	N/A
McDonnell-Douglas REVEAL	640K	all	—	no	no	yes	yes	yes	no	yes	yes	yes	no	no	no	yes	yes	yes	N/A
Advanced systems																			
Software Architecture & Engineering Inc. KES	640K	512K	unlimited[1]	yes	yes	no	yes	no	no	yes	yes	yes	no	no	yes	no	yes	yes	yes
Teknowledge Inc. M.1	192K	all	200	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	no	yes	yes

N/A = Not applicable.

[1] Expandable to 18 matrices.

[2] Total memory limited.

[3] Virtual memory system.

[4] Refers to ability to make fact statements in knowledge base.

Table 1.

640K RAM. Anything less than this and the system would run sluggishly and have to garbage collect rather frequently. When the long-awaited IQLISP compiler and a significantly updated version of this system are released, KES could become one of the most powerful expert system development tools available for a moderate price.

Teknowledge M.1

M.1 is a PROLOG-based knowledge engineering tool from Teknowledge Inc. that offers several advanced features. Like ES/P Advisor, it was written with PROLOG-1, an implementation of Edinburgh PROLOG produced by Expert Systems International.

The knowledge representation language is very full, allowing even the user interactive commands to be called from within a knowledge base. There are various provisions for controlling how the inference engine operates on a knowledge base that add power and efficiency to knowledge systems built with M.1. The environment for user consultations with finished applications is very polished and includes a multiple window panel display, but it requires specific programming to provide the end user with the most helpful features.

Because of its PROLOG heritage, M.1 essentially uses a backward chaining inference engine. You give it a goal and it searches the rule base for rules with that goal as a consequence or *then* part. It then tries to prove the rule's premise true or to make the premise true. To remove a lot of redundant searching, there is a cache for storing conclusions. Normally, M.1 enters results in this cache as it finds them.

However, some facilities allow both the user and the application developer to interact directly with the cache. At any time, a user can call upon M.1 either to save to the disk whatever is currently in the cache or to load a cache file from the disk. The knowledge engineer can also do this from within the knowledge base and can specify certain items to be automatically inserted in the cache. The advantage of this is that the conclusions are there when needed, so M.1 doesn't waste time searching for them in the knowledge base. It is also possible to design knowledge systems so that alternative fact models can be loaded from the disk to try "what if" scenarios by applying the same rule set to different sets of factual values or assumptions. It also makes it possible for very large applications to be broken down into a set of smaller, more manageable modules.

If M.1 doesn't find a value in the cache or in a fact in the knowledge base for an expression it needs to reach its goal, and if there is no provision for calculating this value from rules it already knows, then it will automatically ask the user for the value it needs or use the text provided by the knowledge engineer to pose the question.

M.1's knowledge base language has three types of entries: facts, metafacts, and rules. A metafact is an entry that tells the M.1 inference engine to behave in a certain way. For example, the "question" metafact instructs M.1 to ask the user a question using a specific text to obtain a specific value.

One very useful and powerful feature of this knowledge-based language is its ability to use variables anywhere and in any of the three entries. Thus it is possible to create very efficient knowledge bases. Generic rules can be written in terms of variables and the facts can be supplied like lookup tables for the rules. Not only does this keep the rule base small, but a variety of fact bases often can be used as alternate models with the same rule base. This provides a degree of modularity not often found in knowledge engineering systems.

Another very useful feature of M.1 is its ability to allow you to vary its syntax by defining new operators. Thus it is possible for rule statement expressions to look as much like English phrases as the developer wants. For example, if you have an expression in the knowledge base like "end-of-consultation," it can easily be changed to a more readable form, such as "the end of the consultation," by defining the right prefix, infix, and postfix operators. Normally, you would develop your application in the less friendly form and then, after completing it, define all the operators needed to make it read more like ordinary English.

If you can't share files on PC Network, you're using the wrong file manager.

Be connected. Btrieve.™

Networks can solve problems. But running a single-user file manager can create new ones: Lost updates. Garbled data. Trashed files.

Btrieve™/N offers *safe* multi-user file management that protects your data when sharing files. And eliminates the need to rewrite your application for LANs. Btrieve/N set the file management standard for the industry's most popular networks: Netware, Davong MultiLink, Omninet, PC Net, EtherSeries, Nestar, and NetOne. And now IBM's PC Network.

Fast. Btrieve/N is fast, too. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing, for access speed that won't degrade as your database grows.

Automatic file recovery. Btrieve/N provides automatic file recovery after a system crash. Your Btrieve data always comes back intact.

Fully-relational data management. SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™/N adds report writing capabilities. Xtrieve™/N speeds users through database queries with interactive menus.

For professional programmers. Btrieve/N is the fast, reliable answer for all your application development in BASIC, Pascal, COBOL, C, FORTRAN, and APL. With Btrieve/N, you can develop better network applications. And solve problems, not create new ones.

SC SoftCraft Inc.
P. O. Box #917 Austin, Texas 78766
(512) 346-8380 Telex 358 200

Suggested retail prices: Btrieve, $245; Btrieve/N, $595; Xtrieve, $195; Xtrieve/N, $395; Rtrieve, $85; Rtrieve/N, $175. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X.

CIRCLE 89 ON READER SERVICE CARD

Pascal Users—Save Time & Errors
(with PascalPac™ and tidy™)

For the IBM PC, XT, AT.

A major step toward paperless programming!
Debug on the screen!
Neater programs!

PascalPac™
X-REF creates cross reference table.
X-RAY browses cross reference and program simultaneously.
X-PRNT is a versatile listing program.
X-PEEK browses programs or text files.
One version of PascalPac supports Microsoft and TURBO PASCAL.

tidy™
Program formatter makes a PASCAL program easier to read, understand and modify. **In use for over one year in major companies and programming organizations.** Available in separate versions for Microsoft and TURBO PASCAL.

"**tidy** is a lightning-fast Pascal formatter from Major Software."
"Of the products we've seen, we felt that tidy is the one most programmers will prefer."
PC Tech Journal

PascalPac....**$69;** Separate Modules...**$60;** tidy-Turbo...**$49;** tidy Microsoft...**$69;** Shipping...**$5**

To order: VISA/MasterCard orders. **Call (415) 941-1924.** Or mail check/money order to:
Major Software, 66 Sylvian Way, Los Altos, CA 94022

CIRCLE 14 ON READER SERVICE CARD

Now With Windowing!
$49.95 Basic Compiler

MTBASIC

Features:

Multitasking	**Windowing**
Handles interrupts	**Interactive**
Fast native code	**Compiles quickly**
Floating point	**No runtime fee**

MTBASIC is a true native code compiler. It runs *Byte's* Sept. '81 sieve in 26 seconds; interpreters take over 1400 seconds! Because MTBASIC is multitasking, it can run up to 10 Basic routines at the same time, while displaying ten separate windows. Pop-up/down menus are a snap to implement.

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so *any* system can use windowing), three demonstration programs and a comprehensive manual.

AVAILABLE for CP/M (Z-80), MS-DOS, and PC-DOS systems.

ORDERING: Specify format when ordering. We accept Visa, MC, checks and COD. Send $49.95 plus $3.50 shipping and handling ($10 overseas) to:

SOFTAID, Inc.
P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

CIRCLE 96 ON READER SERVICE CARD

BORED?...
...waiting for C programs to compile and link?

Use C-terp
the complete C interpreter

This is the product you've been waiting (and waiting) for!

Increase your productivity and avoid agonizing waits. Get instant feedback of your C programs for debugging and rapid prototyping. Then use your compiler for what it does best...compiling efficient code ...slowly.

C-terp Features

- Full K&R C (no compromises)
- Complete built-in screen editor-- no half-way house, this editor has everything you need such as multi-files, inter-file move and copy, global searching, auto-indent, tab control, and much more.
- Fast--Linking and semi-compilation are breath-takingly fast. (From edit to run completion in a fraction of a second for small programs.)
- Convenient--Compiling and running are only a key-stroke or two away. Errors direct you back to the editor with the cursor set to the trouble spot.
- Object Module Support -- Access functions and externals in object modules produced by C86 or Lattice C or assembly language. Utilize your existing libraries unchanged!
- Complete Multiple Module Support-- Instant global searches, auto-compile everything that's changed, etc.
- Many more features including batch mode, 8087 support and symbolic debugging.
- Runs on IBM PC, DOS 2.x, 192K and up.
- **Price: $300.00 (Demo $45.00) MC, VISA**

Price of demo includes documentation and shipping within U.S. PA residents add 6% sales tax. Specify C86 or Lattice version.

GIMPEL SOFTWARE
3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

*Trademarks: C86 *(Computer Innovations)*, Lattice *(Lattice Inc.)*, IBM *(IBM Corp.)*, C-terp *(Gimpel Software)*

M.1 now has an interface arrangement that allows functions to be called from an external library through the use of an external metaproposition. Assembly language, or any compiled programming language that conforms to the standards of the supported interface, can be used to prepare external functions. The interface requires an additional simple assembly language routine called *USER*, which forms the bridge to M.1. An object code module, M1AUX.OBJ, is linked with the *USER* module and the external functions to form the M1AUX.EXE file. This file is loaded when M.1 boots, giving it access to the external functions.

This gateway also allows up to eight arguments, either numbers or atoms (strings), to be passed both to and from M.1. Each external function is accessed by an integer that is the function identifier assigned to it. There can be up to 200 functions, each with a unique function identifier. These functions can access external data collection hardware as well as external data bases and even other machinery and computing devices.

The major limitations of M.1 are its speed (which is not overpowering), its 200-rule limit, and its use of only a backward chaining inference engine. But other than speed, none of these are absolute limitations. It is not a problem to design knowledge systems with M.1 that would consist of a network of component files, all 200 rules or less, that directly call and load one another as the need arises.

Such a system has many possible architectures. They range from one where the first knowledge base file is really a discrimination net that determines which of a large number of other expert systems to load to more complex networks of interrelated knowledge files. Also, powerful techniques using recursive procedures, cycles, list processing, and metarules and metapropositions can permit the design of knowledge systems that get some of the same results as forward chaining and other types of inference engines.

All in all, M.1's power is very impressive for its size, even though its price limits it to a rather select clientele.

Texas Instruments Personal Consultant

One expert systems tool we still have not received as this article goes to press is Personal Consultant from TI. It is available for the IBM PC and compatibles as well as the TI Professional. Briefly, it is a LISP-based, backward-chaining expert systems shell of the familiar Mycin type. It offers function key commands and an English-like syntax for rule explanations and allows the inclusion of custom routines written in IQ LISP.

Personal Consultant is available for MS-DOS releases 1.1 through 2.1, requires 512K RAM and, according to the TI brochure, can accommodate up to 400 rules. The price is $3,000. Personal Consultant comes packaged with the latest release of IQ LISP.

Expert Systems products and manufacturers

OPS5+—$3,000
Artelligence Inc.
14902 Preston Rd., Ste. 212-252
Dallas, Texas 75240
(214) 437-0361

ES/P Advisor—$895
Expert Systems International
1150 First Ave.
King of Prussia, Pa. 19406
(215) 337-2300

TIMM-PC—$9,500
General Research Corp.
7655 Old Springhouse Rd.
McLean, Va. 22102
(703) 893-5900

Insight Knowledge System 2—$485
Level Five Research Inc.
4980 S. Highway A-1-A
Melbourne Beach, Fla. 32951
(305) 729-9046

TK!Solver—$399
Lotus/Software Arts
27 Mica Ln.
Wellesley, Mass. 02181
(617) 237-4000

REVEAL—$2,000
McDonnell-Douglas
20705 Valley Green Dr.
Cupertino, Calif. 95104
(408) 446-6000

KES—$4,000
Software Architecture & Engineering
1500 Wilson Blvd., Ste. 800
Arlington, Va. 22209
(703) 276-7910

M.1—$10,000
Teknowledge Inc.
525 University Ave.
Palo Alto, Calif. 94301
(415) 327-6600

Personal Consultant—$3,000
Texas Instruments
P.O. Box 809063
Dallas, Texas 75380-9063
(800) 527-3500

STRUCTURE FOR BASIC

With PROFESSIONAL PROGRAMMING ENVIRONMENT

ELIMINATES LINE NUMBERS

ALLOWS MULTI-LINE CONDITIONALS

PCDOS/MSDOS

BENDORF ASSOCIATES
6006 S. MAIN
P.O. BOX 5910
ROSWELL, NM
88201
505 347-5701
VISA/MASTERCARD

WORKS WITH BASICA INTERPRETER COMPILER

FULL ERROR LOGGING

PROGRAM LISTER

LABELED PROCEDURES

MACROS

SUB-ROUTINES

LIBRARIES

$49.95

CIRCLE 6 ON READER SERVICE CARD

MINNESOTA

SNOBOL4

ONLY $44.95

This exceptional language will amaze you with its versatility. Compatible with main-frame SNOBOL4, use it at work and home. Perform those complicated programming jobs and prototype your exotic ideas quickly. This robust implementation supports large memory model, 8087(if float desired), 32 bit integers, 32K strings. Includes 60 page reference guide and sample programs including ELIZA. You can also get Griswold's et al definitive "green" book or even the source code. Needs IBM PC or 8088/86, ≥128K, IBM or MS DOS.

Guide + 5¼" SSDD diskette	$44.95
Guide + diskette + "green" book	$59.95
"Green" book only	$24.95
Source code and license	$500.00

Prices postpaid. In NY state add sales tax.

☎ (914)271-5855

BERSTIS INTERNATIONAL
P.O. Box 441
MILLWOOD, NY 10546 USA

CIRCLE 88 ON READER SERVICE CARD

ZAS

Software

Development Package

The industry's most sophisticated development tool is now available for the **Z-8000** and the **Z-8** under **CP/M, MSDOS**, and **ISIS.**

Includes:

- ZAS Relocatable Macro Cross Assembler
  -38 Directives, nested macros, etc.
- ZLK Task Builder/Linker
  -resolves CALR's, section oriented
- ZLD User-Modifiable Object Loader
- ZEX Dual Processor Run-Time Support

CP/M TM Digital Research
MSDOS TM Microsoft
ISIS TM Intel Corp

Western Wares
303-327-4898
Box C • Norwood, CO 81423

CIRCLE 31 ON READER SERVICE CARD

NEW!

Advanced Trace86™

Symbolic Debugger & Assembler Combo

- Full-screen trace with single stepping; Even backstepping!
- Write & Edit COM & EXE programs
- Conditional breakpoints (programmable)
- Switch between trace and output screen; Or set up two monitors
- 8087, 80186, 80286, 80287 support
- Write labels & comments on code
- Polish hex/decimal calculator
- and more . . . Priced at $175.00

To order or request more information contact:

Morgan Computing Co., Inc.
P.O. Box 112730, Dallas, TX 75011
(214) 245-4763

CIRCLE 22 ON READER SERVICE CARD

COMPUTER LANGUAGE source code now available in disk formats! (Details on page 4)

Contact:
CL Users Group
COMPUTER LANGUAGE
131 Townsend St.
San Francisco, Calif. 94107

C Users' Group

Over 40 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group
415 E. Euclid • Box 97
McPherson, KS 67460
(316) 241-1065

CIRCLE 62 ON READER SERVICE CARD

Pascal-to-C Translator: Convert UCSD Pascal, MT+ Pascal, and others to K&R C. Handles nested procedures, intrinsic functions, separately compiled units and modules, all data types including long integers. Requires 256K IBM PC/XT/AT. Send for free samples or send us up to 500 lines of Pascal and we will convert it for you for **FREE**. Licensed source and executable code $5,000, conversions 25 cents/line.

TGL Inc.
4400 Sulphur Springs Rd.
Corvallis, OR. 97330
(503)-745-7476

CIRCLE 95 ON READER SERVICE CARD

Finally. BSW-Make.

The Boston Software Works now brings a complete implementation of the Unix "make" facility to MS-DOS. No more recompiling every file in sight after a small edit; no more wondering if you've really rebuilt every module affected by an edit. Just type "make" and BSW-Make automatically builds your product quickly, efficiently and correctly.

BSW-Make supports:

- most compilers and assemblers
- MS-DOS or PC-DOS v2.00 or later
- macros for parameterized builds
- default rules
- MS-DOS pathnames
- any MS-DOS machine (192K minimum)

Only $69.95 postpaid (Mass. residents add 5% sales tax)

The Boston Software Works
120 Fulton Street, Boston, MA 02109
(617) 367-6846

CIRCLE 3 ON READER SERVICE CARD

FORTRAN PROGRAMMERS

Discover why you should be using
F77L
the complete implementation of the ANSI FORTRAN 77 Standard for the IBM PC and compatibles.
If you are serious about your FORTRAN programming, you should be using F77L.
$477

Lahey Computer Systems, Inc.
31244 Palos Verdes Drive West, Suite 243
Rancho Palos Verdes, California 90274
(213) 541-1200
Serving the FORTRAN community since 1969

CIRCLE 15 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
AGS Computers Inc.	1	7
Amber Systems	5	2
Argon Universal Software	85	5
Arity Corp.	20	59
Bendorf & Associates	103	6
Berstis International	103	88
Blaise Computing Inc.	15	1
Borland International	cover IV	4
Boston Software Works (The)	103	3
C Users Group	103	62
C Ware/DeSmet	98	11
CCA Uniworks Inc.	3	60
Computer Helper Industries	91	10
COMPUTER LANGUAGE C Seminar	46,47	97
Creative Programming	71	52
Digitalk Inc.	8	45
Ecosoft	71	9
Edward Ream	91	—
Entelekon	71	40
Essential Software Inc.	64	33
Exsys Inc.	24	54
Forth Inc.	23	30
GKS Associates	81	71
Gimpel Software	83	—
Gimpel Software	101	—
Gracon Services Inc.	22	56
Greenleaf Software Inc.	90	44
HSC Inc.	59	12
Harvard Softworks	93	47
Haventree Software	10	67
Inmos Corp.	25	32
Laboratory Microsystems	56	48
Lahey Computer Systems Inc.	103	15
Lattice Inc.	75	16
Level Five Research Inc.	83	90
Lifeboat Associates	60	87
Lifeboat Associates	82	85
LISP Company (The)	91	65
Lisp Lab Inc. (The)	69	58
mbp Software & Systems Technology	65	53
MIX Software	89	24
Major Software	101	14
Manx Software Systems	cover III	69
Megamax Inc.	81	13
MicroTech Research Inc.	73	75
Micromotion	56	64
Microsoft Corp.	6	—
Miller Microcomputer Services	95	20
Mindbank Inc.	70	63
Morgan Computing Co	103	22
Mystic Canyon Software	76	46
Mytech Software Inc.	66	19
Next Generation Systems	94	61
Northwest Computer Algorithms	85	39
Novix	67	73
OES	66	98
Pascom Computing	cover II	—
Phoenix Computer Products Corp.	77	18
Polytron Corp.	38	34
Programmer's Connection	97	17
Programmer's Shop (The)	48	35
Programmer's Shop (The)	62	78
Rational Systems Inc.	96	72
Realia Inc.	58	76
Relational Database Systems Inc.	45	77
SLR Systems	83	28
Shaw Laboratories Inc.	18	79
Softaid Inc.	101	96
SoftCraft Inc.	100	89
Softfocus	67	—
Software Horizons	69	25
Softway Inc.	34	29
Solution Systems	93	42
Solution Systems	93	91
Sota Computing Systems Ltd.	37	36
Sparry Software Labs	91	66
Speedware	85	41
Spruce Technology Corp.	17	26
Systems Guild	19	27
Systems Peripherals Consultants	69	38
TGL Inc.	103	95
Tardis Software	92	82
Technisoft	80	83
Turbo Power Software	72	57
UniPress	19	21
Vermont Creative Software	68	8
Wendin	88	43
Western Wares	103	31
Wizard Systems	67	86
Woodchuck Industries	81	55
Workman & Associates	81	68

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

ADVERTISE in the November issue of COMPUTER LANGUAGE

Algorithms issue

Reservation deadline: Sept. 9th
Contact:
Carl Landau
COMPUTER LANGUAGE
131 Townsend St.
San Francisco, Calif. 94107
(415) 957-9353

AZTEC The Most Powerful C

for the IBM AT • MACINTOSH • MS DOS • CP/M-80 • ROM APPLICATIONS
IBM PC/XT • APPLE // • CP/M-86 • TRSDOS • CROSS DEVELOPMENT

Why Professionals Choose Aztec C

AZTEC C compilers generate fast, compact code. AZTEC C is a sophisticated development system with assemblers, debuggers, linkers, editors, utilities and extensive run time libraries. AZTEC C is documented in detail. AZTEC C is the most accurate and portable implementation of C for microcomputers. AZTEC C supports specialized professional needs such as cross development and ROM code development. MANX provides qualified technical support.

AZTEC C86/PRO
— for the IBM AT and PC/XT

AZTEC C86/PRO provides the power, portability, and professional features you need to develop sophisticated software for PC DOS, MS DOS AND CP/M-86 based microsystems. The system also supports the generation of ROM based software for 8088/8086, 80186, and 80286 processors. Options exist to cross develop ROM code for 65xx, 8080, 8085, and Z80 processors. Cross development systems are also available that target most micro computers. Call for information on AZTEC C86/PRO support for XENIX and TOPVIEW.

POWERFUL — AZTEC C86/PRO 3.2 outperforms Lattice 2.1 on the DHRYSTONE benchmark 2 to 1 for speed (17.8 secs vs 37.1) while using 65% less memory (5.8k vs 14k). The AZTEC C86/PRO system also compiles in 10% to 60% less time and supports fast, high volume I/O.

PORTABLE — MANX Software Systems provides **real** portability with a family of compatible AZTEC C software development systems for PC DOS, MS DOS, CP/M-86, Macintosh, CP/M-80, APPLE //+, //e, and //c (NIBBLE - 4 apple rating), TRSDOS (80-MICRO - 5 star rating), and Commodore C64 (the C64 system is only available as a cross compiler - call for details). AZTEC C86/PRO is compatible with UNIX and XENIX.

PROFESSIONAL — For professional features AZTEC C86/PRO is unparalleled.

- Full C Compiler (8088/8086 - 80186 - 80286)
- Macro Assembler for 8088/8086/80186/80206
- Linkage Editor with ROM support and overlays
- Run Time Libraries - object libraries + source DOS 1.x; DOS 2.x; DOS 3.x; screen I/O; Graphics; UNIX I/O; STRING; simulated float; 8087 support; MATH; ROM; CP/M-86
- Selection of 8088/8086, 80186, or 80286 code generation to guarantee best choice for performance and compatibility
- Utility to convert AZTEC object code or libraries to Microsoft format. (Assembly + conversion takes less than half the time as Microsoft's MASM to produce MS object)
- Large memory models and sophisticated memory management
- Support products for graphics, DB, Screen, & ...
- ROMable code + ROM support + separate code and data + INTEL Hex Converter
- Symbolic Debugger & Other Utilities
- Full Screen Editor (like Vi)
- CROSS Compilers are available to APPLE //, Macintosh, CP/M-80, TRSDOS, COMMODORE C64, and ROM based 65xx, and 8080/8085/Z80
- Detailed Documentation

AZTEC C86/PRO-AT (configured for IBM AT - options for 8088/8086)	$500
AZTEC C86/PRO-PC/XT (configured for IBM PC/XT - options for 80186/80286)	$500

AZTEC C86/BAS includes C compiler (small model only), 8086 MACRO assembler, overlay linker, UNIX, MATH, SCREEN, and GRAPHICS libraries, debugger, and editor.

AZTEC C86/BAS	$199
AZTEC C86/BAS (CP/M-86)	$199
AZTEC C86/BAS (DOS + CP/M-86)	$299
UPGRADE to AZTEC C86/PRO	$310
C-TREE Database with source	$399
C-TREE Database (object)	$149

CROSS COMPILERS

Cross Compilers for ROM, MS DOS, PC DOS, or CP/M-86 applications.

VAX - > 8086/80xxx cross	$5000
PDP-11 - > 8086/80xxx cross	$2000

Cross Compilers with PC DOS or CP/M-86 hosts are $750 for the first target and $500 for each additional target. Targets: 65xx; CP/M-80; C64; 8080/8085/Z80; Macintosh; TRSDOS; 8086/8088/80186/80286; APPLE //.

AZTEC C68K
— for the Macintosh

For power, portability, and professional features AZTEC C68K-c is the finest C software development system available for the Macintosh.

The AZTEC C68K-c system includes a 68000 macro assembler, a linkage editor, a source editor, a mouse based editor, a SHELL development environment, a library of UNIX I/O and utility routines, full access and support of the Macintosh TOOLBOX routines, debugging aides, utilities, make, diff, grep, TTY simulator with upload & download (source supplied), a RAM disk (for 512K Mac), a resource maker, and a **no royalty** license agreement. Programming examples ar included. (Over 600 pages of documentation).

AZTEC C68K-c requires a 128K Macintosh, and two disk drives (frugal developers can make do with one drive). AZTEC C68K supports the 512K Macintosh and hard disks.

AZTEC C68K-c (commercial system)	$500
AZTEC C68K-p (personal system)	$199
AZTEC C68K-p to AZTEC C68K-c upgrade	$310
Mac C-tree database	$149
Mac C-tree database with source	$399
Lisa Kit (Pascal to AZTEC C68k object converter)	$ 99

AZTEC C65
— for the APPLE //

"...The AZTEC C-system is one of the finest software packages I have seen..." NIBBLE review, July 1984.

The only commercial C development system available that runs native on the APPLE II+, IIc, and IIe, the AZTEC C65 development system includes a full floating point C compiler compatible with UNIX C and other MANX AZTEC C compilers, a 6502 relocating assembler, a linkage editor, a library utility, a SHELL development environment, a full screen editor, UNIX I/O and utility subroutines, simple graphics, and screen functions.

AZTEC C65 (Apple DOS 3.3)	$199
AZTEC C65/PRO (Apple DOS + ProDos) (call for availability)	$350

AZTEC C II/PRO
— for CP/M-80

The first member of the AZTEC C family was the CP/M-80 AZTEC C compiler. It is "the standard" compiler for development on CP/M-80. The system includes the AZTEC C II C compiler, an 8080 assembler, a linkage editor, an object librarian, a full library of UNIX I/O and utility routines, CP/M-80 run time routines, the SMALL library (creates modules less than 3K in size), the fast linker for reduced development times, the ROM library, RMAC and M80 support, library source, support for DRI's SID/ZSID symbolic debugger, and more.

AZTEC C II/PRO	$349
AZTEC CII/BAS	$199
C-TREE Database with source	$399
C-TREE Database in AZTEC object form	$149

AZTEC C80
— for TRSDOS (Radio Shack Model III & 4)

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen." 80-Micro, December, 1984, John B. Harrell III

This sytem has most of the features of AZTEC C II for CP/M. It is perhaps the best software development system for the Radio Shack Model III and IV.

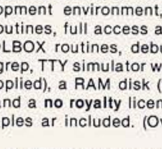
AZTEC C80 model 3 (no floating point)	$149
AZTEC C80 model 4 (full)	$199
AZTEC C80/PRO (full for model 3 and 4)	$299

To order or for information call:

800-221-0440

(201) 530-7997 (NJ and outside U.S.A.). Or write: MANX SOFTWARE SYSTEMS, P.O. Box 55, Shrewsbury, N.J. 07701.

MANX



SHIPPING INFORMATION - Standard U.S. shipment is UPS ground (no fee). In the U.S. one day shipment is $20, two days is $10. Canadian shipment is $10. Two days shipment outside the U.S. is by courier and is freight collect.

TRS 80 RADIO SHACK TRS DOS is a trademark of TANDY.
APPLE DOS MACINTOSH is a trademark of APPLE.

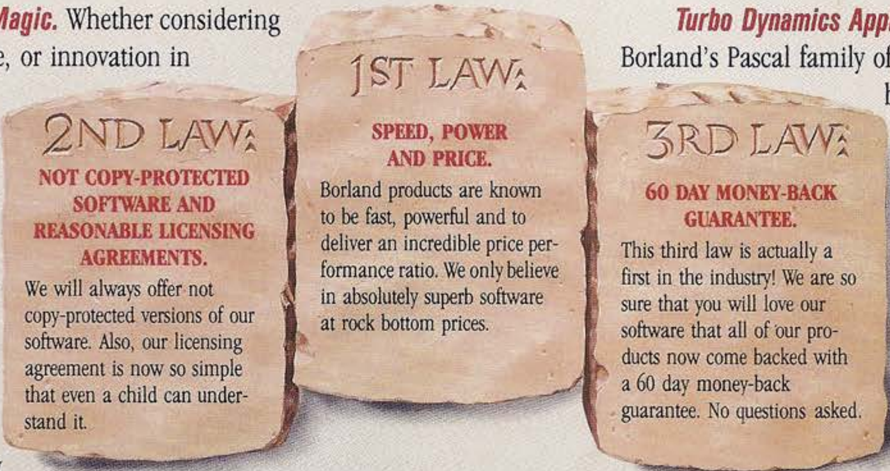
For Technical Support
(Bug Busters) call: 201-530-6557

CIRCLE 69 ON READER SERVICE CARD

PC WEEK'S PRODUCT OF THE YEAR
PC MAGAZINE'S AWARD FOR TECHNICAL EXCELLENCE

Borland Introduces the Laws of *TURBO DYNAMICS*™

Laws That Work Like Magic. Whether considering technological excellence, or innovation in areas such as pricing, not copy-protection, licensing agreements, site licenses, 60 day money-back guarantee—Borland is clearly recognized as the software industry leader. The following three laws of "*Turbo Dynamics*"™ exemplify our pledge for excellence.



1ST LAW:

SPEED, POWER AND PRICE.

Borland products are known to be fast, powerful and to deliver an incredible price performance ratio. We only believe in absolutely superb software at rock bottom prices.

2ND LAW:

NOT COPY-PROTECTED SOFTWARE AND REASONABLE LICENSING AGREEMENTS.

We will always offer not copy-protected versions of our software. Also, our licensing agreement is now so simple that even a child can understand it.

3RD LAW:

60 DAY MONEY-BACK GUARANTEE.

This third law is actually a first in the industry! We are so sure that you will love our software that all of our products now come backed with a 60 day money-back guarantee. No questions asked.

Turbo Dynamics Applies to Turbo Pascal. Borland's Pascal family of products is growing by leaps and bounds. You can now join hundreds of thousands of users and enter the world of Turbo Pascal programming. And remember, all three laws of *Turbo Dynamics* apply to all Borland products.

TURBO PASCAL™ *$69.95*

The industry standard. With more than 350,000 users worldwide Turbo Pascal is the industry's de facto standard. Turbo Pascal is praised by more engineers, hobbyists, students and professional programmers than any other development environment in the history of microcomputing. And yet, Turbo Pascal is simple and fun to use. **Free spreadsheet** included on every Turbo disk with ready-to-compile source code. ***Options:*** We offer the exciting Binary Coded Decimal (BCD) option for your business applications as well as an 8087 option for your number-crunching applications at a very low charge. Please refer to the coupon. ***Portability.*** Turbo Pascal is available today for most computers running PC-DOS, MS-DOS, CP/M-80 or CP/M-86. ***Jeff Duntemann, PC Magazine:*** "In its simplicity it achieves an elegance that no other language compiler has ever displayed."

TURBO DATABASE TOOLBOX™ *$54.95*

The Turbo Database Toolbox is the perfect complement to Turbo Pascal. It contains a complete library of Pascal procedures that allows you to sort and search your data and build powerful applications. It's another Borland set of tools that will give the beginning programmer the expert's edge. ***Get started right away: free database!*** Included on every Toolbox disk is the source code to a working data base which demonstrates how powerful and easy to use our search system, Turbo-Access, really is. Modify it to suit your individual needs or just compile it and run. ***Remember, no royalties!***

TURBO GRAPHIX TOOLBOX™ *$54.95*

High resolution monochrome graphics for the IBM PC.

The Turbo Graphix Toolbox will give even a beginning programmer the expert's edge. It's a complete library of Pascal procedures and functions. Tools that will allow you to draw and hatch pie charts, bar charts, circles, rectangles and a full range of geometric shapes. Procedures that will save and restore graphic images to and from disk. And much, much, more. You may incorporate part or all of these tools in your programs and yet we won't charge you any royalties. Best of all, these functions and procedures come complete with commented source code on disk ready to compile.

TURBO TUTOR™ *$34.95*

From start to finish in 300 pages. Turbo Tutor is for everyone from novice to expert. Even if you've never programmed before Turbo Tutor will get you started right away. ***A must.*** You'll find the source code for all the examples in the book on the accompanying disk ready to compile. Turbo Tutor might be the only reference on Pascal and programming you'll ever need.

NOT COPY-PROTECTED

TURBOPASCAL FAMILY

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

Pascal 3.0	$ 69.95	______
Pascal w/8087	$109.90	______
Pascal w/BCD	$109.90	______
Pascal w/8087 & BCD	$124.95	______
Turbo Database Toolbox	$ 54.95	______
Turbo Graphix	$ 54.95	______
Turbo Tutor	$ 34.95	______

*These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Carefully Describe your Computer System!

Mine is: ___ 8 bit ___ 16 bit
I Use: ___ PC-DOS ___ MS-DOS
___ CP/M 80 ___ CP/M 86

My computers' name/model is: ______

The disk size I use is:
☐ 3½" ☐ 5¼" ☐ 8"

60 DAY MONEY-BACK GUARANTEE

Amount: (CA 6% tax) ______
Payment: VISA MC BankDraft Check
Credit Card Expir. Date: ____/____
Card #: ______

Name: ______
Shipping Address: ______
City: ______
State: ______ Zip: ______
Telephone: ______

COD's and Purchase Orders WILL NOT be accepted by Borland. California residents: add 6% sales tax.
Outside USA: add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

F20

BORLAND INTERNATIONAL

4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

Copyright 1985 Borland International BI-1011

Turbo Pascal, Turbo Database Toolbox, Turbo Graphix Toolbox, Turbo Tutor and Turbo Dynamics are trademarks of Borland International, Inc.

CIRCLE 4 ON READER SERVICE CARD